Titre: Title:	Arc Routing Problems for Road Network Maintenance
Auteur: Author:	Ingrid Marcela Monroy Licht
Date:	2015
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Monroy Licht, I. M. (2015). Arc Routing Problems for Road Network Maintenance [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1856/

UNIVERSITÉ DE MONTRÉAL

ARC ROUTING PROBLEMS FOR ROAD NETWORK MAINTENANCE

INGRID MARCELA MONROY LICHT

DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIÆ DOCTOR

(GÉNIE INDUSTRIEL)

AOÛT 2015

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

ARC ROUTING PROBLEMS FOR ROAD NETWORK MAINTENANCE

présentée par : MONROY LICHT Ingrid Marcela

en vue de l'obtention du diplôme de : Philosophiæ Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. GENDREAU Michel, Ph. D., président

M. LANGEVIN André, Ph. D., membre et directeur de recherche

M. AMAYA GUIO, Ciro Alberto, Ph. D., membre et codirecteur de recherche

M. ROUSSEAU Louis-Martin, Ph. D., membre et codirecteur de recherche

M. GAMACHE Michel, Ph. D., membre

M. EGLESE Richard, Ph. D., membre externe

DEDICATION

To my Parents Astrid and Fernando,

To my Grandmother Cecilia

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Dr. André Langevin for his valuable guidance and support. He was always available and patient for my questions. I owe him my gratitude for supporting me in difficult moments.

I would also like to thank my supervisor Dr. Ciro Alberto Amaya for sharing expertise, valuable guidance and encouragement extended to me.

I am also extremely grateful to my supervisor Dr. Louis-Martin Rousseau for his help, valuable guidelines and suggestions.

I wish to thank the jury members of my dissertation for taking the time to read this work.

I owe my thanks to the staff at Department of Mathematics and Industrial Engineering and at Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT).

Finally, I would like to thank my family for their support and for being there for me through ups and downs.

RÉSUMÉ

Cette thèse présente deux problèmes rencontrés dans l'entretien des réseaux routiers, soit la surveillance des réseaux routiers pour la détection de verglas sur la chaussée et la reprogrammation des itinéraires pour les activités de déneigement et d'épandage de sel. Nous représentons ces problèmes par des modèles de tournées sur les arcs. La dépendance aux moments et la nature dynamique sont des caractéristiques propres de ces problèmes, par conséquence le cas de surveillance des réseaux routiers est modélisé comme un problème de postier rural avec fenêtres-horaires (RPPTW), tandis que le cas de la reprogrammation utilise des modèles obtenus à partir des formulations de problèmes de tournées sur les arcs avec capacité.

Dans le cas du problème de surveillance, une patrouille vérifie l'état des chemins et des autoroutes, elle doit principalement détecter le verglas sur la chaussée dans le but d'assurer de bonnes conditions aux chauffeurs et aux piétons. Étant donné un réseau routier et des prévisions météo, le problème consiste à créer une tournée qui permette de détecter opportunément le verglas sur les rues et les routes. L'objectif poursuivi consiste à minimiser le coût de cette opération.

En premier, on présente trois formulations basées sur la programmation linéaire en nombres entiers pour le problème de surveillance des réseaux qui dépend du moment et deux méthodes de résolution: un algorithme de coupes et un algorithme heuristique appelé *adaptive large neighborhood search* (ALNS). La méthode exacte inclut des inéquations valides tirées du problème du voyageur de commerce avec fenêtres-horaires et aussi du problème de voyageur du commerce avec contraintes de précédence. La méthode heuristique considère deux phases: en premier, on trouve une solution initiale et après dans la deuxième phase, l'algorithme essaie d'améliorer la solution initiale en utilisant sept heuristiques de destruction et deux heuristiques de réparation choisies au hasard. La performance des heuristiques est évaluée pendant les itérations. Une meilleure performance correspond à une plus grande probabilité de choisir une heuristique.

Plusieurs tests ont été faits sur deux ensembles d'exemplaires de problèmes. Les résultats obtenus montrent que l'algorithme de coupes est capable de résoudre des réseaux avec 104 arêtes requises et des fenêtres-horaires structurées par tranches horaires ; l'algorithme peut aussi

résoudre des réseaux avec 45 arêtes requises et des fenêtres-horaires structurées pour chaque arête requise. Pour l'algorithme ALNS, différentes versions de l'algorithme sont comparées. Les résultats montrent que cette méthode est efficace parce qu'elle est capable de résoudre à l'optimalité 224 des 232 exemplaires et de réduire le temps de calcul significativement pour les exemplaires les plus difficiles.

La dernière partie de la thèse introduit le problème de la reprogrammation de tournées sur les arcs avec capacité (RCARP), lequel permet de modéliser la reprogrammation des itinéraires après une panne d'un véhicule lors de la phase d'exécution d'un plan initial des activités de déneigement ou d'épandage de sel. Le planificateur doit alors modifier le plan initial rapidement et reprogrammer les véhicules qui restent pour finir les activités. Dans ce cas, l'objectif poursuivi consiste à minimiser le coût d'opération et le coût de perturbation. La distance couverte par les véhicules correspond au coût d'opération, cependant une nouvelle métrique est développée pour mesurer le coût de perturbation. Les coûts considérés sont des objectifs en conflit. On analyse quatre politiques à la phase de re-routage en utilisant des formulations de programmation linéaire en nombres entiers.

On propose une solution heuristique comme méthode pour résoudre le RCARP quand les coûts d'opération et de perturbation sont minimisés en même temps et quand une réponse rapide est nécessaire. La méthode consiste à fixer une partie de l'itinéraire initial et après à modifier seulement les itinéraires des véhicules les plus proches de la zone de l'interruption de la tournée du véhicule défaillant. La méthode a été testée sur des exemplaires obtenus d'un réseau réel. Nos tests indiquent que la méthode peut résoudre rapidement des exemplaires avec 88 arêtes requises et 10 véhicules actifs après la panne d'un véhicule.

En conclusion, la principale contribution de cette thèse est de présenter des modèles de tournées sur les arcs et de proposer des méthodes de résolution d'optimisation qui incluent la dépendance aux temps et l'aspect dynamique. On propose des modèles et des méthodes pour résoudre le RPPTW, et on présente des résultats pour ce problème. On introduit pour la première fois le RCARP.

Trois articles correspondant aux trois principaux chapitres ont été acceptés ou soumis à des revues avec comité de lecture: "The rural postman problem with time windows" accepté dans *Networks*, "ALNS for the rural postman problem with time windows" soumis à *Networks*, and

“The rescheduling capacitated arc routing problem” soumis à *International Transactions in Operational Research*.

ABSTRACT

This dissertation addresses two problems related to road network maintenance: the road network monitoring of black-ice and the rescheduling of itineraries for snow plowing and salt spreading operations. These problems can naturally be represented using arc routing models. Timing-sensitive and dynamic nature are inherent characteristics of these problems, therefore the road network monitoring is modeled as a *rural postman problem with time windows* (RPPTW) and in the rescheduling case, models based on capacitated arc routing formulations are suggested for the rerouting phase.

The detection of black-ice on the roads is carried out by a patrol to ensure safety conditions for drivers and pedestrians. Specific meteorological conditions cause black-ice on the roads; therefore the patrol must design a route covering part of the network in order to timely detect the black-ice according to weather forecasts. We look for minimum-cost solutions that satisfy the timing constraints.

At first, three formulations based on mixed integer linear programming are presented for the timing-sensitive road network monitoring and two solution approaches are proposed: a cutting plane algorithm and an *adaptive large neighborhood search* (ALNS) algorithm. The exact method includes valid inequalities from the *traveling salesman problem* (TSP) *with time windows* and from the *precedence constrained* TSP. The heuristic method consists of two phases: an initial solution is obtained, and then in the second phase the ALNS method tries to improve the initial solution using seven removal and two insertion heuristics. The performance of the heuristics is evaluated during the iterations, and therefore the heuristics are selected depending on their performance (with higher probability for the better ones).

Several tests are done on two sets of instances. The computational experiments performed show that the cutting plane algorithm is able to solve instances with up to 104 required edges and with time windows structured by time slots, and problems with up to 45 required edges and time windows structured by each required edge. For the ALNS algorithm, several versions of the algorithm are compared. The results show that this approach is efficient, solving to optimality 224 of 232 instances and significantly reducing the computational time on the hardest instances.

The last part of the dissertation introduces the *rescheduling capacitated arc routing problem* (RCARP), which models the rescheduling of itineraries after a vehicle failure happens in the execution of an initial plan of snow plowing or salt spreading operations. A dispatcher must quickly adjust the remaining vehicles and modify the initial plan in order to complete the operations. In this case we look for solutions that minimize operational and disruption costs. The traveled distance represents the operational cost, and a new metric is discussed as disruption cost. The concerned objectives are in conflict. Four policies are analyzed in the rerouting phase using mixed integer linear programming formulations.

A heuristic solution is developed to solve the RCARP when operational and disruption costs are minimized simultaneously and a quick response is needed. The idea is to fix part of the initial itinerary and only modify the itinerary of vehicles closer to the failure zone. The method is tested on a set of instances generated from a real network. Our tests indicate that the method can solve instances with up to 88 required edges and 10 active vehicles after the vehicle breakdown.

In short the main contribution of this dissertation is to present arc routing models and optimization solution techniques that consider timing-sensitive and dynamic aspects. Formulations and solution methods with computational results are given for the RPPTW, and the RCARP is studied for the first time here.

Three articles corresponding to the main three chapters have been accepted or submitted to peer review journals: “The rural postman problem with time windows” accepted in *Networks*, “ALNS for the rural postman problem with time windows” submitted to *Networks*, and “The rescheduling capacitated arc routing problem” submitted to *International Transactions in Operational Research*.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ALNS	Adaptive Large Neighborhood Search
ATSP-TW	Asymmetric Traveling Salesman Problem with Time Windows
CARP	Capacitated Arc Routing Problem
CARPTW	Capacitated Arc Routing Problem with Time Windows
CPP	Chinese Postman Problem
CPPTW	Chinese Postman Problem with Time Windows
GRASP	Greedy Randomized Adaptive Search Procedure
LP	Linear Program
MIP	Mixed integer Program
PC-ATSP	Precedence Constrained Asymmetric Traveling Salesman Problem
RCARP	Rescheduling Capacitated Arc Routing Problem
RWIS	Road Weather Information System
RPP	Rural Postman Problem
RPPTW	Rural Postman Problem with Time Windows
TSP	Traveling Salesman Problem
VND	Variable Neighborhood Descend
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

CHAPTER 1 INTRODUCTION

Road network maintenance, especially in winter is a significant challenge to most governments and transportation agencies in North America. An indication of the magnitude of this operation is related to the high costs that it implies. In Ontario, the total expenditures in highway winter maintenance reached $171 million in the 2013 fiscal year (Office of the Auditor General of Ontario, 2015). Michigan Department of Transportation spent $103 million for fiscal year 2013 (Slone, 2014). The New Jersey Department of Transportation reported that it spent a record $138 million to keep state roadways clear of snow and ice for 2013 (Slone, 2014). The Pennsylvania Department of Transportation, which had $189.2 million budgeted for the 2013-14 winter, spent $284 million (Slone, 2014). Edmonton, the city which spends more on snow removal than any other city in western Canada, estimated a budget of $50.4 for all the winter road maintenance operations for 2013 (Rodrigues, 2013). Bob Dunford, Director of Roadway Maintenance in Edmonton, listed the expenses of winter maintenance in the city (interview given to Rodrigues (2013)): “The plowing we do is about $19 million and sanding is about $11 million, removal is about $6.1 million and then you’ve got a snow storage site at about $2 million. Also, we have 1100 km of sidewalks that we are responsible for, and that’s $7 million a year we spend on that.”

Winter road maintenance includes all the operations that aim at the removal or reduction of snow and ice on roadways providing safe winter driving conditions and safe sidewalks for pedestrians. The roads and highways must be kept cleared of snow and ice on a reasonably time basis. Providing an efficient winter road service is a responsibility of municipalities, but in many cases they outsource the winter maintenance operations to private-sector contractors.

Winter maintenance authorities are constantly seeking for technology-based solutions such as advanced road weather information systems for monitoring localized weather and road surface conditions, automated vehicle location and Global Positioning System for tracking fleet operations and performance (Fu et al. 2009). However to obtain a great benefit from these technologies, the integration with algorithms that support the decisions and plans in operational, strategic and tactical level of road winter maintenance is necessary. Decision-making at the supervisory level of winter maintenance operations are often complex and constrained by time

and resources. For example for salt spreading, Eglese (1994) stated that “if a road is treated too early, then the salt may be washed away if rain is falling or blown away by the wind before the temperature drops to freezing conditions. If the road is treated too late, then ice may have already started to form and the road will be dangerous for traffic traveling on the road before it has been treated”.

Arc routing is an operational research area that has contributed with models and algorithms to assist road network maintenance managers and operators to make more structured decisions. The use of operational research in this field could results in substantial saving, improved mobility, and reduced environmental and social impacts (Liu et al. 2014). However, for many years the focus of arc routing in winter maintenance has been in static problems where all data are assumed known before the routes are constructed and do not change afterward. But the reality of winter road maintenance is really complex and has a dynamic nature.

The new information technologies mentioned above, and other technological advances in communication systems allow the exploration of real time information for dynamic routing and scheduling. This favors getting more realistic representation of the operations and enhancing the performance of decision systems in the area of routing.

Few works have attempted to address the **time-sensitive** and **dynamic** nature of winter maintenance operations. This Dissertation focuses on these two complex issues of road network maintenance. The objective is to develop decision support arc routing models and optimization solution methods for two cases: road network monitoring and rescheduling of salt spreading and snow plowing operations. The first case considers the time sensitivity due to weather forecasts in the schedule of the monitoring of black-ice on roads, and the second case considers a dynamic component, where given an initial schedule for spreading or plowing operations, it must be modified when a vehicle breakdown occurs.

Road network monitoring: In the region of “Estrie” in the Province of Quebec, from mid-October to mid-December a patrol daily checks the state of roads and must detect black ice on roads in a timely fashion in order to avoid pedestrian falls or automobile accidents. Once the patrol has noticed a possible risk for safe mobility, it reports the incident to service centers in charge of road signs and special works. The planner of the route has access to short-term weather forecasts and a characterization of roads with likelihood of black-ice formation. The planner must

match the temperature conditions in different zones with the probability of the presence of black-ice and plan a circuit for the patrol.

Rescheduling of salt spreading - snow plowing operations: Snow removal and salt spreading, the most common activities in winter maintenance, are often performed for each storm event on a repetitive basis over the same routes until safe mobility conditions are achieved (Campbell, Langevin, and Perrier, 2014). In practice master schedules are usually constructed by solving deterministic capacitated arc routing instances based on average demands predicted by the weather conditions. The master schedules should be executed as they were established if no unexpected events occur. However, disruptions may occur, especially in winter, which may interrupt the master plans. When a disruption occurs, routes should be quickly revised to minimise the negative impact it may cause and to ensure the quality of service. The case of adjusting an initial itinerary after one or more vehicle breakdowns during the execution stage is considered in this research.

This dissertation consists of four main chapters: Chapter 2 presents the literature review. In Chapter 3 an exact approach is proposed to solve the *rural postman problem with time windows* (RPPTW); the article of Chapter 3 has been accepted for publication in *Networks*. In Chapter 4 a heuristic method is presented to solve larger instances of the same problem; the corresponding article has been submitted to *Networks*. In Chapter 5 the *rescheduling capacitated arc routing problem* is introduced considering operational and disruption costs; the article of Chapter 5 has been submitted to *International Transactions in Operational Research*. A more detailed outline of the thesis is given in the following section.

1.1 Thesis Outline

At first, the contributions in the literature on time-sensitive and dynamic problems in arc routing problems, especially in road winter maintenance applications are reviewed. This review concludes that more research is needed on time-sensitive and dynamic winter maintenance arc routing problems.

Then, Chapters 3 and 4 present the case of road network monitoring. In the first article (Chapter 3) the monitoring of roads for black-ice detection is modeled as a RPPTW. In the literature it is known that the polyhedral theory is very successful in solving postman problems,

especially the NP-hard postman problems (Tan et al. 2013). This approach is explored after presenting several mathematical formulations of the problem. A cutting plane algorithm is proposed as a solution method and tested on sets of instances adapted from the literature and on the real network of the “Estrie” region.

The second article (Chapter 4) presents a metaheuristic as a solution technique to the monitoring of road network when large instances are considered. The technique is based on the *adaptive large neighborhood search* metaheuristic, which is one of the most successful metaheuristics for solving complex routing problems (Ropke and Pisinger, 2006a). Several versions of the metaheuristic combining different removal operators are compared. The performance of the metaheuristics is evaluated in comparison to the solution of the first approach.

Regarding to dynamic issues in road winter maintenance operations, the third article (Chapter 5) deals with the vehicle failures during the execution phase of salt spreading or snow plowing operations. In dynamic routing problems all or part of the information are revealed or updated as the routes are executed. Hence, the planners must react to events that occur in real time. This work considers a reschedule as response to the disruption in a *capacitated arc routing problem* itinerary. The objective is to study different policies in the rerouting phase considering operational and disruption costs. Formulations based on mixed integer programming are presented and a solution strategy is discussed when operational and disruption costs are minimized simultaneously.

CHAPTER 2 LITERATURE REVIEW

An important area for the applications of arc routing is concerned with road network maintenance. In particular, there are various logistic operations that deal with maintenance of roads in winter. Spreading chemicals and abrasives, plowing roadways and sidewalks, loading snow into trucks, and ice control are some examples of operations which are key activities to maintaining the safety and the mobility in cities and rural areas. The complex operations, the infrastructure constraints, especially in urban areas, and the dynamic nature of the operating conditions make the road network maintenance a challenge for many governments.

Arc routing problems in the context of road network maintenance differ from other arc routing applications in a number of important ways due to climate, level of service, network complexity and size, traffic conditions, turn restrictions, synchronization of operations, policy decisions and others. Therefore, arc routing models and solutions that consider practical complexities of the problem are of enormous worth.

Campbell and Langevin (2000) present a brief history of roadway snow and ice control in the U.S. from 1862 to 1996 and a survey of early works addressing arc routing in the same frame from 1970 to 1994. In addition, a general description of two successful software packages for roadway snow and ice control is presented. The first one CASPER, developed in India incorporates multiple objectives in designing routes via a penalty function. The objectives included in the software consider: meeting specific service level (time limit for routes), minimizing deadhead travel, and maintaining class continuity when priorities are given. The second software package is GeoRoute Municipal, which has been developed in Canada; its module “route manager” is its optimization system. It works dividing the region of study into sectors, and then for each to optimize the routes for a number of scenarios. The survey provides evidence of the wide gap between theory and practice, therefore authors encourage contributions in the area of route optimization for winter road maintenance.

Perrier et al. (2006a, 2006b, 2007a, 2007b) present a comprehensive review of models and algorithms developed for the variety of winter road maintenance operations. This work is divided into 4 surveys. The first one focuses on optimization models and solution algorithms for the

design for spreading and plowing. The second one discusses system design problems for snow disposal operations. The last two address vehicle routing, depot location, and fleet sizing models for winter road maintenance.

Perrier et al. (2012) provide a survey of recent optimization models and solution methodologies for the routing of spreading operations. They present a detailed classification scheme for spreader routing models developed over the past 40 years. They emphasize that the new models demonstrate impressive capacities to include more issues of the real complexity, the use of more sophisticated hybrid solutions strategies and consideration of more comprehensive models that integrate vehicle routing with other strategic winter maintenance problems. However, they note that there is still a large gap between state-of-the-art models and actual implementations.

Campbell et al. (2014) present the most recent survey on operational research methods on snow plow routing and a case study on implementation of route optimization for snow plowing. They report the works not covered in the survey of Perrier et al. (2007b). This review documented the trend from a heuristic approach to mathematical programming-based approaches, as well as efforts to include more issues of the real complexity in snow plowing routing. They conclude that even when in the last decade there has been some impressive progress in snow plow routing research, similar progress has not occurred in implementing route optimization for winter road maintenance because it seems that models are generally still not comprehensive enough to consider all that needs to be included, and the mathematical programming-based models do not have the ease of use required by operating personnel.

Eglese et al. (2014) provide detailed background on arc routing for the salt spreading. The survey shows that while early work has used simple constructive heuristics, more recently various metaheuristics algorithms have been developed for salt spreading applications. The authors note that there are few works of exact solution methods being used in real cases because a reasonable amount of computing time is required. Hence solution methods tend to rely on heuristic approaches. Two works are highlighted: Letchford and Eglese (1998) and Tagmouti et al. (2007), which are particularly relevant to the types of model and constraints found in spreading applications.

Road network monitoring and road marking are other activities of road network maintenance; however in arc routing problems they have not received as much attention as snow plowing or spreading chemicals for winter.

The road network monitoring aims at maintaining the safety and the visibility of the roads and highways by a timely detection of the various incidents occurring on it. This problem was firstly modeled by Marzolf et al. (2006) as a *periodic capacitated arc routing problem* for the case where vehicles must inspect all the categorized road segments of a network over a two-week horizon. During a shift, a vehicle may have to leave the planned route to answer an emergency call and it may not be able to complete its planned itinerary. The authors rebuilt the routes using a mathematical linear formulation which selects pre-determined routes in order to maximize the number of passages on class 1 arcs. Later, Monroy et al. (2013) studied the same problem, but this time the objective is to build the initial plan minimizing the traveled cost and fulfilling frequencies of services on the roads classified into three categories. They present a mixed-integer program formulation and for larger instances they propose a heuristic solution method that works in two phases; first an assignation of arcs to shift is done, and then a *rural postman problem* is solved for each shift.

In the Province of Quebec, road markings have to be painted or repainted every year. The Ministry of Transport uses a fleet of special vehicles to mark the roads and also tank trucks to meet the marking vehicles and replenish them. Amaya et al. (2007) introduce the *capacitated arc routing problem with refill point*, the problem consists in simultaneously determining the routes of marking vehicles and refilling vehicles that minimize the total cost. An integer linear programming model is presented and a cutting plane method to solve it. Later, Amaya et al. (2010) present a route-first cluster-second heuristic procedure for an extension of the problem considering multiple loads; in this version the refilling vehicle does not have to return to the depot each time it meets the marking vehicles.

The time-sensitivity of the operations and the dynamic nature of the context are two important features on road winter maintenance noted in previous surveys. Eglese et al. (2014) and Perrier et al. (2012) affirm that the timing of operations is crucial to achieve the desired level of service in road winter maintenance. On the other hand, a real-time routing in road network maintenance is necessary to respond dynamically not just to atmospheric conditions and forecasts, but also to

equipment breakdowns, traffic congestion and accidents, all of which are more common in winter driving (Perrier et al., 2012). The survey from "synthesis report on winter highways operations" (Transportation Research Board, 2005) that included 22 prominent winter road maintenance agencies in North America highlights the use of dynamic routes in practice as 72% of the agencies indicated that they dynamically change routes.

Table 2.1 summarizes the works on time-sensitive arc routing problems and dynamic arc routing problems in road network maintenance presented in the surveys.

Table 2.1: Synthesis of works in road network maintenance considering time-sensitive and dynamic context

Problem	**Authors**	**Real application**	**Problem characteristics**	**Solution method**
CARPTW*	Eglese (1994)	Spreading operations	Multi-depots Wide time windows	Two-phase heuristic
RPP** with deadline classes	Letchford and Eglese (1998)	Spreading operations	Deadline classes	Cutting-plane approach
CARPTW*	Golbaharan (2001)	Plowing routing	Multi-depots Time windows	Column generation
CARPTW*	Razmara (2004)	Plowing routing	Time windows	Column generation
Rescheduling for Periodic CARP*	Marzolf et al. (2006)	Road network monitoring	Frequencies of service Reschedule of routes	Mathematical formulations solved using CPLEX 8.0
CARP*** with dynamic information	Handa et al. (2005)	Spreading operations	Requirements and demands change during the operation	Memetic algorithm
CARP*** with time dependent service cost	Tagmouti et al. (2007)	Spreading operations	Time-dependent service cost	Column generation
CARP*** with time dependent service cost	Tagmouti et al. (2010)	Spreading operations	Time-dependent service cost	Variable neighborhood descent
Dynamic CARP*** with time dependent service cost	Tagmouti et al.(2011)	Spreading operations	Dynamic version Time-dependent service cost	Variable neighborhood descent

CARPTW*: Capacitated arc routing problem with time windows
RPP**: Rural postman problem
CARP***: Capacitated arc routing problem

From Table 2.1, it can be seen that most works that consider timing-sensitive arc routing problems have focused on the capacitated case and column generation is the approach most used as the solution method. There are only three works that deal with dynamic aspects, in these cases heuristics and metaheuristics are proposed to solve the problem.

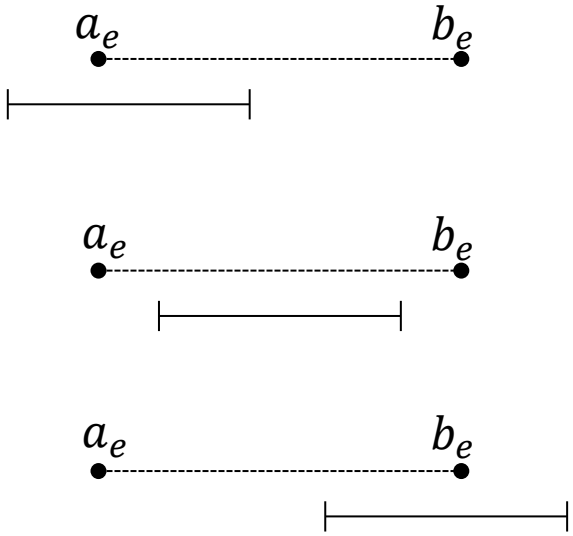
Next sections correspond to contribution on time-sensitive (time windows) and dynamic arc routing problems where details of most of the works in Table 2.1 are presented.

2.1 Arc routing problems with time windows

Arc routing problems focus on solving routing when the demands for services are located on edges or arcs of a network. Arc routing is the counterpart of vehicle routing, and addresses cases like waste collection where demands correspond to quantities to be collected in the streets, or salt spreading for ice clearance in winter where deliveries must be done over roads.

When each customer e specifies a period of time, called a "time window" in which the service must occur, the conventional arc routing problems become arc routing problems with time windows. A time interval $[a_e, b_e]$ is specified for each customer, where $a_e \geq 0$ indicates the earliest arrival time and $b_e > 0$ indicates the latest arrival time. Most of the arc routing problems with time windows consider the case when the service must start between the earliest and latest arrival time of the time windows. However there are other types of time windows: i) the service must finish no later than the latest time; in this case every time window begins at time zero and only deadlines are given to customers. ii) The service must be carried out during the interval of the time window. This is typical in cases such as trash collection where the collection must respect some schedules in large cities, or such as street sweeping where the activity must be done during some specific time because of parking restrictions. The Figure 2.1 presents the different types of windows.

Other classification of routing with time windows considers soft and hard time windows. In the soft case, the vehicles are allowed to violate time windows but a penalty cost is incurred. On the other hand, in hard time windows a feasible solution must satisfy the time windows constraints for all the services and vehicles may wait at a node for service.



a. The service must finish before the latest time.

b. The service must be executed during the time interval

c. The service must start between the earliest and the latest time

Figure 2.1: Types of time windows

Time dependent arc routing problems can be seem as arc routing problems with soft time windows when the service cost is minimal within a given time interval and then increases linearly on both sides of the interval.

The following sections present a brief description of the arc routing problems and a summary of the research that has been done considering timing-sensitive features.

2.1.1 The Chinese postman problem

This problem was first suggested by the Chinese mathematician Kwan (1962). Formally, given a connected graph $G = (V, E)$ where V is a set of vertices and E is a set of undirected edges, with distances on the edges, the problem is to find a tour, which passes through every edge in E at least once, starting and finishing at the same vertex, and in the shortest possible way. When the underlying graph is completely directed or completely undirected, the *Chinese postman problem* (CPP) can be solved in polynomial time (Christofides, 1973; Edmonds and Johnson, 1973). However when the underlying graph is mixed, the problem becomes NP-hard (Papadimitriou, 1976).

2.1.1.1 The Chinese postman problem with time windows

This problem is an extension of the CPP, where one vertex of the graph G is designed as the depot vertex, and the tour must start and finish at that vertex. In addition, time intervals are

introduced so that earliest and latest time constraints are specified for the start of service of each edge. The inclusion of time windows constraints makes the problem NP-hard in all the cases (Dror, 2000).

The CPP *with time windows* (CPPTW) is studied at first by Wang and Wen (2002). They consider a mixed linear programming model of a directed CPP and incorporate the time-constraints into the model. The formulation traces how to travel the network structure explicitly. The problem is defined on a directed graph $G = (V, A)$ where $V = \{v_i | \ i = 1, \dots, n\}$ is a vertex set and $A = \{(v_i, v_j) | \ (v_i, v_j) \neq (v_j, v_i), \forall \ v_i, v_j \in V \}$ is an arc set. Let T_i^k be the time that the postman visits vertex v_i at k^{th} iteration, assuming that the postman starts and finishes the tour at vertex 1, then T_1^1 means the postman's starting time at vertex 1, and $T_1^{K+1}, K \geq 1$ is the completion time of the tour when K is large enough. D_{ij} is the distance (time) from vertex v_i to vertex v_j and it is assumed that the triangle inequality holds for the distance measure. Let x_{ij}^k be a binary variable equal to 1 if at k^{th} iteration the postman traverses from vertex v_i to vertex v_j, and equal to 0 otherwise. The value of K must be given under the condition of $K \geq \max_{(i,j)}\{\sum_k x_{ij}^k\}$. Assuming that when a postman must traverse an arc more than once, the postman will deliver the mail when he traverses the arc at the first time, the model can be formulated as follows:

$$\text{minimize } T_1^{k+1} - T_1^1 \tag{2.1}$$

s.t.:

$$T_i^k - T_1^k \geq D_{1i} x_{1i}^k \quad \forall \ (1, v_i) \in A, k = 1, \dots, K \tag{2.2}$$

$$T_1^{k+1} - T_i^k \geq D_{i1} x_{i1}^k \quad \forall \ (v_i, 1) \in A, k = 1, \dots, K \tag{2.3}$$

$$T_j^k - T_i^k \geq D_{ij} x_{ij}^k \quad \forall \ (v_i, v_j) \in A, i, j \neq 1, \ k = 1, \dots, K \tag{2.4}$$

$$\sum_{(v_i, v_j) \in A} x_{ij}^k = \sum_{(v_j, v_i) \in A} x_{ji}^k \quad \forall \ i = 1, .., n, \ k = 1, \dots, K \tag{2.5}$$

$$\sum_k x_{ij}^k \geq 1 \quad \forall \ (v_i, v_j) \in A, k = 1, \dots, K \tag{2.6}$$

$$a_i \leq T_i^1 \leq b_i \quad \forall \ i = 2, .., n \tag{2.7}$$

$$x_{ij}^k \in \{0,1\} \quad \forall \ (v_i, v_j) \in A, k = 1, \dots, K \tag{2.8}$$

The objective function looks for minimizing the total time for the postman to finish the tour. Constraints (2.2), (2.3), and (2.4) ensure the continuity of time. In (2.5) the condition that vertices must be symmetric is ensured. Constraints in (2.6) ensure that each arc must be passed at least once. Constraints (2.7) impose that postman has to arrive vertex i between time interval $[a_i, b_i]$ to deliver the mails. The variables are restricted to be 0-1 integer in (2.8).

Note that this model does not consider waiting time once the postman starts the mail delivering. However, the authors modified the model a little to ensure the existence of the solution in the cases where waiting time is necessary and they employ the concept of fuzzy set theory when time constraints are not certain.

This model includes a strong assumption: the "iteration" variable is a simple circuit. The problem is formulated as a circuit sequence such that each vertex in an iteration associates with only one starting time and in every circuit the depot vertex is included. Although the authors found an optimal value of K based on the even-degree properties of a directed CPP, this result is only valid for the case where the time windows can be fixed. They determined the bounds of the time intervals to get feasible solutions. However, if an arbitrary time interval is given, there can be no solution to this model.

Aminu and Eglese (2006) studied the undirected case and modelled the problem using constraint programming. Two different formulations are proposed. The first formulation approaches the problem directly and the second transforms the problem to an equivalent *vehicle routing problem with time windows* (VRPTW). In these formulations it has been assumed that the time to travel over an edge is equal to the cost of travelling over the edge.

First formulation (F1):

Initially an arbitrary edge is added to represent the depot node, therefore the total number of edges in the graph is $N + 1$, if N is the number of the original edges in the graph. Three sets of decision variables are defined: i) $Edge = \{E_1, \dots, E_N, E_0\}$ is the set of variables where E_i indicates where $e_i \in E$ comes in the ordering for service and E_0 represents the order for servicing the edge depot (the domain of the $Edge$ variables is $\{1, \dots, N + 1\}$). ii) a set of binary variables to indicate in which direction the edge is served; for each edge $e_i \in E$, z_i determines the direction that the edge is serviced (for $e_i = (v_p, v_q)$, $z_i = 0$ if the edge is serviced in the direction $v_p \rightarrow v_q$

and $z_i = 1$ if the edge is serviced in the direction $v_q \rightarrow v_p$). iii) a set of variables $cost_i$ for $e_i \in E$ represent the cost (or time) to finish serving e_i. Let $cost_{ij}$ be the cost of the shortest path from the node where $e_i \in E$ was served last to the node where service starts for $e_j \in E$. Let $acost_j$ be the actual given cost to service the length of $e_j \in E$, and let $cost_j$ be the cost to finish serving $e_j \in E$. Then, for edge $e_j$succeeding $e_i$in the solution, the relationship $cost_i + cost_{ij} + acost_j \leq cost_j$ holds. In addition, let a_i and b_i be the earliest and latest time to serve edge e_i, and let u_cost be the upper bound for the cost of the complete tour. F1 for the CPPTW is stated as follows:

minimize $cost_{N+1}$ (2.9)

s.t.:

$cost_i + cost_{ij} + acost_j \leq cost_j \; \forall \; i \neq j, e_i, e_j \in E$, when e_j follows e_i (2.10)

$a_i \leq cost_i \leq b_i \; for \; i = 1, \dots, N$ (2.11)

$0 \leq cost_N \leq u\_cost$ (2.12)

$Edge :: [1, \dots, N+1]$ (2.13)

$alldifferent\,(Edge)$ (2.14)

$E_0 = N + 1$ (2.15)

The objective function minimizes the time to complete service on all the edges of the graph starting and finishing at v_0. The cost of the current partial path is computed in (2.9), time windows constraints are defined in (2.10) - (2.12). The domain of the $Edge$ variables is given in (2.13) and (2.14) imposes that these variables must be all different. In (2.15), the variable representing the depot node is constrained to be served the last.

The second formulation (F2)

The second formulation is based on a graph transformation inspired by Pearn et al. (1987) where each edge of the graph is replaced with three nodes called the side and middle nodes. Using this approach the problem is transformed into a *vehicle routing problem* (VRP). The use of the middle node ensures that the three nodes representing an edge are serviced consecutively. However, the authors dispense the middle nodes and use just the side nodes to represent each

edge, and adding additional constraints. Demands of each edge side node are half of the demand on the edge they represent. The transformed problem may now be formulated as a special case of a *clustered travelling salesman problem with time windows* where each member of the cluster must be visited within its time windows before visiting a node in a different cluster. The clusters are formed by the side node pairs and additional constraints are added to ensure that the side nodes are consecutive node in any route.

F1 found optimal solutions for problems with up to 15 edges and tight time windows. F2 was tested on a set of problems with up to 69 edges. Optimal solutions were found quickly when the time windows are tight. The results also show that as the time windows are made wider and the number or feasible solutions increases, some problems are not solved to optimality within a reasonable computing time.

2.1.1.2 The Chinese postman problem with time-dependent travel time

In this problem the travel time on an arc depends on the starting time to travel it. Sun et al. (2013) studied the test sequence optimization in hybrid automaton (Springintveld et al. 2001), where the delay time of transition from state s_i to s_j is a function $D_{ij}(t_i)$ of the arrival time t_i at s_i. The authors model the problem on a directed network $G = (V, A)$ with $D_{ij}(t_i)$ as the time dependent travel time of arc $(v_i, v_j) \in A$. As each state s_i corresponds to the vertex v_i in V, and each transition from s_i to s_j corresponds to the arc (v_i, v_j) in A, the optimal test sequence checking all transitions on the hybrid system is equivalent to a minimum *time dependent* CPP-tour that traverses all the arcs in time dependent network G. The problem is formulated as an integer programming model which uses the iteration variable introduced by Wang and Wen (2002) to trace the tour. To solve the problem the authors propose a cutting plane heuristic algorithm. They present cutting planes that can be separated polynomially using a maximum flow algorithm. In addition two upper bound heuristics are also designed to terminate the cutting plane procedure whenever the current LP solution is fractional and violates no inequality. The algorithm is tested on a set of generated instances with up to 25 vertices and 50 arcs. The computational results show that the lower bound obtained by adding cutting planes improves the LP relaxation bound of the original formulation for all instances. The gap between the lower bound and the best upper bound for all the tests ranges between 0.90% and 31.75%.

Later, Sun et al. (2015), propose an integer programming approach, an extension of the previous formulation. The new formulation does not assume that every cycle in the graph must visit the depot. They use two sets of constraints: the first part has a strong combinatorial structure, which is linear and refers to the routing; the second part is related to time-dependent travel time and is not linear. In the case when all the travel times are piecewise functions of the starting time, a linearization is provided. The formulation is solved with a cutting plane algorithm. The algorithm was tested on a real-world and several randomly generated instances. For the real instance with 27 vertices and 27 arcs the algorithm finds a gap between the lower and upper bounds is less than 11.80% in 425 s. For the set of generated instances with between 10 to 25 vertices and 20 to 60 arcs, and piece functions with 2 to 4 times intervals, and fluctuation interval $[-R, R]$ from $[-10,10]$ to $[-30,30]$. The results indicate that the face defining and valid inequalities proposed play an important role when the number of time intervals and the scale of fluctuation become larger.

2.1.1.3 Time dependent Chinese postman problem with time windows

Sun et al. (2011) consider a variant of the CPPTW, where the travel time and service time on arcs depend on the starting time. The travel and service times are piecewise linear functions of time.

The problem is defined on a directed graph $G = (V, A)$, where V is the vertex set, and A is the arc set. Each arc $(v_i, v_j) \in A$ has an associated time window $[a_{ij}, b_{ij}]$, with $a_{ij}, b_{ij} \in \mathbb{Z}^+ \cup \{0\}$. The travel time $D_{ij}(t_i) \in \mathbb{Z}^+$ of an arc $(v_i, v_j) \in A$ depends on the starting time t_i. Similarly, let $S_{ij}(t_i) \in \mathbb{Z}^+$ be the time dependent service time of arc $(v_i, v_j) \in A$ with starting time t_i. The problem looks for finding the minimum total travel time tour starting at the given depot vertex $v_0 \in V$ and starting time t_0 and passing through each $(v_i, v_j) \in A$ at least once, such that the completion time of servicing each arc in A is in its associated time window.

The authors present a graph transformation method through which the *time dependent* CPPTW can be reformulated as a 0/1 integer linear programming without any timing constraint. The problem is transformed into a *generalized rural postman problem*. The algorithm is tested on five sets of randomly generated instances, where $|V|$ ranges from 30 to 50, and $|A|$ ranges from 50 to 130. They were able to solve to optimality instances with up to 100 arcs within 15 minutes plus

the time of transforming the graph. In addition, they tested four sets of instances to find how the width of time windows can affect the computational time.

2.1.2 The rural postman problem

This problem was introduced by Orloff (1976). Formally, the *rural postman problem* (RPP) is defined on an undirected connected graph $G = (V, E)$, where V is the vertex set and E is the edge set. A subset R of the edges are required. Each edge e has a cost $c_e \geq 0$. The RPP consists of finding a minimum cost tour in G traversing every edge in R at least once. The RPP historically arose in rural mail delivery; however applications of the RPP take place in contexts where some edges of a graph must be serviced by an uncapacitated vehicle.

By reducing the NP-hard Hamiltonian cycle problem to the RPP, Lenstra and Kan (1976) showed RPP to be NP-hard, except when $R = E$, in which case the RPP becomes the CPP.

2.1.2.1 The RPP with deadline classes

The RPP *with deadline classes* is presented by Letchford and Eglese (1998). In this problem the set of arcs are partitioned into small number of classes according to priority, with each class having its own deadline on service. Formally the set R of required edges is partitioned into $\{R^1, \dots, R^p\}$ and services for each class k of edges $(k = 1, \dots, p)$ must be completed by time T^k. The problem is formulated on an undirected graph as an integer linear programming model. The authors have proposed several classes of valid inequalities which exploit the structure of the problem. They used the dual cutting-plane method (Nemhauser and Wolsey, 1988) to solve the problem: an initial LP relaxation is solved and then each time that a violated inequality is identified, it is added to the LP, and the LP is solved using the dual simplex method. When no more violated inequalities can be found, branch-and-bound is invoked to obtain integrality. The algorithm was tested on a set of 10 instances: 5 problems from Corberán and Sanchis (1994) were adapted and the value of p was set to 1 and 2 for each of the problems. The problems have between 22 and 67 required edges and between 3 and 6 connected components. The cutting plane algorithm showed a good performance as all the instances were solved to optimality.

2.1.2.2 The time-dependent rural postman problem

Applications involving scheduling with time-dependent processing time (Alidaee and Womer, 1999; Sundararaghavan and Kunnathur, 1994) motive this problem. The travel (or service) time of each arc depends on time, that is, the travel time of an arc depend on the time interval during which the arc is traversed, and the postman is not required to cover every arc in the network, only a subset of arcs. A formal definition is presented by Tan et al. (2013). Let $G = (V, A)$ be a directed graph, where V is the vertex set and A is the arc set, which includes a subset of required arcs A_R that must be serviced. Each required arc in A_R is associated with a travel time and a service time, while the arcs not in A_R have travel time only. Both the travel time and the service time are time-dependent piecewise functions. Let $tt_{ij}(t_i)$ and $st_{ij}(t_i)$ denote the time-dependent travel time function and the service time function, where t_i is the time at the beginning of travel along an arc or the service time on an arc (v_i, v_j). A postman is required to service the arcs in A_R and is located at the depot vertex from which to start and end the service tour. The postman is allowed to wait along the tour and must start after a given time t_0. The *time-dependent* RPP consists of finding a tour servicing all of the required arcs with a minimum cost with respect to the time-dependent travel time and the service time.

Tan and Sun (2011) and Tan et al. (2013) present the version that considers only time-dependent travel times and propose an arc-path formulation and strong valid inequalities. The authors note that the constant travel time assumption in other timing sensitive arc routing problems never holds on the time dependent network. Thus the transformation methods which use the shortest path algorithm have as sub-problem the *time dependent shortest path problem*, which has been proved to be NP-hard (Orda and Rom, 1990). They propose an integer linear programming: an arc-path formulation for the problem with a constraint set divided into two parts. The first part defines the polytope of the arc-path alternation sequence and the second part is closely related to time-dependent travel time. The service time functions are considered as piecewise functions, and are linearized. Based on the polyhedral results, a cutting plane algorithm was proposed as solution method. The algorithm was tested on two sets of randomly generated instances with up to 50 vertices and with up to 50 arcs; the travel time is treated as the step function with 3 and 4 intervals, and the percentage of required arcs ranges from 10% to 30%. The computational results show that for all 42 test instances the method solved instances up to 25

vertices and 50 arcs. The relative gap between the best feasible solution and the lower bound is 3.16%for all the instances on average.

2.1.3 Capacitated arc routing problem

This problem was introduced for the directed case by Golden and Wong (1981), and later (Belenguer and Benavent, 1991) formulated the problem for an undirected graph. It generalizes the Chinese postman and rural postman. Given an undirected graph $G = (V, E)$ with V as the set of vertices and E as the set of edges. A subset R of edges are required. In addition, let $c_e \geq 0$ be the edge cost, $d_e \geq 0$ be the edge demand for every $e \in E$, and Q be the vehicle capacity. The *capacitated arc routing problem* (CARP) consists of determining a minimum cost traversal of all edges in R, so that each vehicle starts and finishes at the depot vertex $v_0 \in V$, and the total demand of all edges serviced by any particular vehicle does not exceed its capacity Q. The CARP defined on two specific graphs is not NP-hard, but on the other cases is NP-hard (Busch, 1991).

2.1.3.1 The CARP with time windows

The CARP *with time windows* (CARPTW) is defined as the classical CARP with the extra requirement that the service of each demand edge must begin within some pre-specific time window.

Given an undirected connected graph $G = (V, E)$, where $c_e \geq 0$ is the cost of traversing the edge $e = (i, j)$, and $d_e \geq 0$ is the demand of the edge. A number of vehicles, each of capacity Q are placed at the node depot v_0. Let t_e denote the time needed to traverse edge e, and let each demand edge have a time windows $[a_e, b_e]$, in which service of the edge must start. The problem consists of finding tours such that i) All edges with $d_e > 0$ are serviced, ii) Vehicle capacities are respected, iii) Service of each demand edge start within the time windows of that edge, and iv) Total cost is minimized.

Eglese (1994) firstly presents an application in routing for winter gritting: local authorities treat the roads by spreading a de-icing agent on them, multiple depot locations and limited vehicle capacities are considered, and roads with different priorities implies that some roads must be treated within two hours and other within four hours of the start of gritting. This problem can be considered as CARPTW, where the time windows are rather wide. A two-phase heuristic method is proposed as solution method: first the optimal solution of an unconstrained CPP for

the network considering just category 1 roads is found, depot locations are specified and routes are defined, second a simulated annealing algorithm attempts to improve the current solution.

Mullaseril (1997) studies the problem of managing a fleet of trucks for distributing feed in a large livestock ranch in Arizona. The ranch produces cattle. The cattle are kept over a large area in approximately 500 pens with rectangular shape. The pens are arranged in rows by a network of paved and dirt roads. The feed type, volume and feeding time for each pen may vary from day to day because there is a constant movement of cattle in and out of the yard. A route may be stipulated to deliver the exact demand to several pens, but weighing inaccuracies may force it to only partially supply the last pen and that pen would need to have feed delivery from more than one route. Thus, the feed-yard allows split-delivery. In the problem case there is a further consideration. A vehicle may traverse the arcs at two different speeds-discharging speed and dead heading speed.

The author models the feed delivery problem for the cattle ranch as a collection of *capacitated rural postman problem with time windows and split delivery*. The livestock ranch is represented as a connected mixed graph, where the set of requirements are arcs (because the design of the delivery trucks). The author presents heuristic algorithms for obtaining fast solutions for this class of problems. Also he presents solution strategies for obtaining tight lower bounds to the optimal solution and optimal solutions to some of the real life split delivery problems with time windows.

In addition, the author presents a transformation of this arc routing setting into an equivalent node routing problem where the number of nodes is the same as the number of required arcs in the original arc routing problem. The problem in the equivalent graph of nodes is decomposed and solved with a column generation approach. The master problem is formulated as a *set covering problem* based on the work of Desrosiers et al. (1995). The sub-problem is modeled as a *shortest path problem with resource constraints* and solved using dynamic programming algorithm as an extension of the work of Desrochers (1988). This approach shows to be successful on instances up to 55 nodes.

Gueguen (1999) describes an integer linear programming model for the undirected CARPTW and another transformation into a VRPTW, but without numerical results. The author presents the only “direct model” for an undirected CARPTW, which is presented as follows:

It is known that an optimal solution to the undirected CARP always traverses each edge at most once per direction in each route. The example of Figure 2.2 shows that this property is not true when time windows are considered. In the example, all costs and traverse times are equal to one, there is not service time considered, and the capacity is not restricted for a vehicle that must start and finish at the depot (black node) and visit all the edges in their time intervals. From the example, the only solution to the problem with a single vehicle is: starts at the depot, serves in this order the edges 5, 1, 4, 2, 3 and finishes at the depot. The vehicle traverses edge 3 six times.

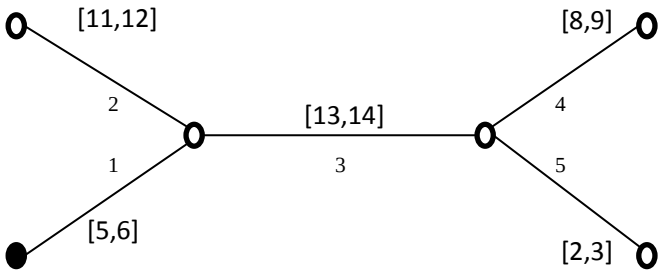


Figure 2.2: Example of time windows in an undirected graph
(Source: Gueguen (1999))

Based on the previous analysis, Gueguen (1999) proposed a model making $(m + 1)$ copies of each edge, if the number of edges of the graph is equal to m. This number is an upper bound considering that in the worst case an edge e will be traversed:

- Once for serving each of the other $(m - 1)$ edges
- Once for serving the edge e
- Once for coming back to the depot.

Having a maximum of $(m + 1)$ times.

Three types of decision variables are defined: traverse, service, and time variables

$x_{i\alpha j\beta}^{k}$ =1 if vehicle k traverse the copy β of edge j immediately after copy α of edge i, and 0 otherwise. These variables are only created when the final node from edge i is the same as the initial node from edge j.

s_i^k =1 if vehicle k services the edge i, and 0 otherwise.

$t_{i\alpha}^{k}$ is the time to start traversing copy α of edge i by vehicle k.

The CARPTW formulation is as follows:

$$\text{minimize } \sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{\propto=1}^{m+1}\sum_{j=1}^{m}\sum_{\beta=1}^{m+1} c_{p_i}^{k} x_{i\alpha j\beta}^{k} + \sum_{k=1}^{K}\sum_{i=1}^{m} c_{s_i} s_i^k \tag{2.16}$$

s.t.:

$$\sum_{j=1}^{m}\sum_{\alpha=1}^{m+1} x_{j\alpha i\beta}^{k} = \sum_{j=1}^{m}\sum_{\alpha=1}^{m+1} x_{i\beta j\alpha}^{k} \quad \forall\, i = 1,\dots,m\ \ \beta = 1,\dots,m+1\ \ k = 1,\dots,K \tag{2.17}$$

$$\sum_{k=1}^{K} s_i^k = 1\ \ \forall\, i = 1,\dots,m \tag{2.18}$$

$$\sum_{i=1}^{m} q_i s_i^k \le Q\ \ \forall\, k = 1,\dots,K \tag{2.19}$$

$$s_i^k \le \sum_{j=1}^{m}\sum_{\alpha=1}^{m+1} x_{i1j\alpha}^{k} \quad \forall\, i = 1,\dots,m \quad k = 1,\dots,K \tag{2.20}$$

$$T_{i\alpha}^{k} + t_{p_i} - C\left(1 - x_{i\alpha j\beta}^{k}\right) \le T_{j\beta}^{k} \quad \forall\, i = 1,\dots,m\ \ j = 1,\dots,m\ \ \beta = 1,\dots,m+1$$
$$\alpha = 2,\dots,m+1,\ \ k = 1,\dots,K \tag{2.21}$$

$$T_{i1}^{k} + t_{p_i} + t_{s_i} - C\left(1 - x_{i1j\beta}^{k}\right) \le T_{j\beta}^{k}\ \ \forall\, i = 1,\dots,m\ \ j = 1,\dots,m\ \ \beta = 1,\dots,m+1$$
$$k = 1,\dots,K \tag{2.22}$$

$$T_{i1}^{k}\ \ge a_i\ \ \forall\, i = 1,\dots,m \quad k = 1,\dots,K \tag{2.23}$$

$$T_{i1}^{k} \le b_i\ \ \forall\, i = 1,\dots,m \quad k = 1,\dots,K \tag{2.24}$$

$$x_{i\alpha j\beta}^{k} \in \{0,1\}\ \ \forall\, i,j = 1,\dots,m \quad \alpha,\beta = 1,\dots,m+1\ \ k = 1,\dots K \tag{2.25}$$

$$s_i^k \in \{0,1\}\ \ \forall\, i = 1,\dots,m \quad k = 1,\dots,K \tag{2.26}$$

$$t_{ik} \in \mathbb{R}^{+}\ \forall\ i\ \in\ A\,,\ k = 1,\dots m \tag{2.27}$$

The objective is to minimize the traverse and service cost. It is assumed that if a service is done in an edge, it is done in the copy 1 of that edge. The constraints (2.17) are conservation flow, the constraints (2.18) and (2.19) make sure that the each edge is serviced, and the vehicle capacity is not exceeded. Constraints (2.20) make a vehicle traverses the copy 1 of an edge if the vehicle serve that edge. Constraints (2.21)-(2.24) set the accumulated times for all copies of the edges and satisfy the time windows when edges are serviced. C is a large constant with value greater than the longest time route. The three sets of variables are defined in (2.25)-(2.27).

As the author noted, the model is not practical, then he proposed a transformation to node routing problem based on the transformation presented by Mullaseril and Dror (1996).

Golbaharan (2001) studies a *multi-depot* CARPTW in the context of snow removal. Every snow plow starts from a depot and returns to the same depot. The problem is formulated as a linear integer programming model; indeed as a *constrained set covering problem*. The objective function in this case minimizes the total cost of the routes and the penalty for using extra snow plows. A column generation method is implemented to solve the problem. The master problem includes the constraints on the number of snow plows available at each depot and the constraints which guarantee that a required road segment is serviced. The sub-problem contains the time window constraints and network flow constraints. The master problem is solved by the dual simplex method, and the sub-problem for every depot is formulated as a *shortest path problem with time windows* associated with the edges in the network, and it is solved with a label-setting algorithm. Computational experiments were conducted on real-life instances involving 7 depots, 21 snow plows, 362 nodes, and 707 required edges.

Razmara (2004) presents a real problem of the Swedish National Road Agency on snow removal routing for homogeneous snowplows; in this case every segment in the network must be plowed in its associated time windows and the routes must start from an end at the same depot. The case is formulated as a linear integer programming problem and solved using a Dantzig-Wolfe decomposition scheme. The master problem is formulated as a *constrained set covering problem*. The sub-problems are resource constrained *shortest path problems*, where time is the resource. The sub-problems are solved by a label-setting algorithm. An integer solution to the master problem is found by a greedy algorithm or a variable reduction procedure.

Later, Wøhlk (2005) provides two mathematical models for the undirected CARPTW, one based on constructing a node duplicated network on which an integer linear programming is built and one based on a transformation into an equivalent node routing problem, the VRPTW. Several heuristics and a dynamic programming algorithm combined with simulated annealing, called DYPSA, are proposed. The average DYPSA performance is about 8% above lower bounds. The author also presents a column generation method to get tight lower bounds.

Reghioui et al. (2007) suggest a *greedy randomized adaptive search procedure* (GRASP) with path relinking for the undirected CARPTW. Two constructive heuristics are used in the GRASP

algorithm: randomized path-scanning heuristic and randomized route first – cluster second heuristic. Local search (OR-OPT, SWAP, and 2-OPT) is used to improve each solution. Path relinking is used as an intensification strategy working on a small pool of elite solutions collected during the GRASP. Computational results showed that the algorithm found 17 optimal solutions (including 4 new ones) on the set of 24 instances proposed by Wøhlk (2005) and the gap to lower bounds is less than 1%.

Johnson and Wøhlk (2009) propose two column generation method and a heuristic approach. The master problem is a large *set partitioning problem*, and the sub-problem finds feasible routes with respect to both capacity and time windows. Two methods are presented to solve *the set partitioning problem*: a two-phase method and an iterative method. The heuristic approach generates only tours that are good in some pre-specified sense and subsequently use the presented methods for solving a set partitioning problem based on these columns. The method is tested on a set of 20 instances adapted from the so-called Eglese instances for the classical CARP. The instances contain from 51 to 190 required edges. The iterative method is superior to the two-phase method both regarding computation time and ability to solve hard instances.

The case where the time windows are soft and violating these implies some extra cost is studied by Afsar (2010). A Dantzing-Wolfe decomposition and column generation approach to solve the problem optimally is presented. The sub-problem is a *non-elementary capacitated shortest path problem*. Computational results are reported on a set of modified instances of Johnson and Wøhlk (2009). The problems have up to 40 nodes and 69 required edges and linear and symmetric penalty costs were considered.

2.1.3.2 Capacitated arc routing problem with time-dependent service cost

Tagmouti et al. (2007) study the directed *capacitated arc routing problem with time-dependent service cost* inspired on winter gritting operations, where the timing of an intervention is crucial; if the intervention is too early or too late, the cost in material and time increases. In this case the cost of service depends on the time of beginning of service, indeed, the cost is a piecewise linear function of time. The problem consists of finding a set of routes that serve all required arcs in the graph at least cost (sum of travel cost and service cost) with the constraint that vehicles are not allowed to wait along their route and must be back at the depot by a given deadline. A column generation algorithm is proposed to solve the problem where the master

problem is a *set covering problem* and the sub-problems are *time-dependent shortest path problems with resource constraints*. The method is tested on a set of instances derived from Solomon's problems of the VRPTW (Solomon, 1987). Instance with up to 40 required arcs were solved to optimality.

Later, Tagmouti et al. (2010) propose a *variable neighborhood descent* (VND) heuristic for solving the problem. Two initial solutions are constructed with an insertion heuristic and an adaption of the savings heuristic (Clarke and Wright, 1964). The VND is then applied to each solution to improve them. The structure of the neighborhoods manipulates an arc or sequence of arcs. They tested the performance of this algorithm on a set of instances adapted from the CARP instances of Golden et al. (1983), Li (1992), and Li and Eglese (1996). The algorithm behaved appropriately and showed to be fast and competitive when compared with the adaptive multi-star local search algorithm of Ibaraki et al. (2005) which is designed for the VRP with soft time windows.

2.2 Dynamic arc routing problems

Several real-life routing problems have been studied in a static context, where it is assumed that all data about the problem are known in advance. However a number of technological advances have made possible that the information available to the planner be updated during the execution of the routes. In this case, part or all of the input is unknown and revealed dynamically during the execution of the plan.

Although there is a fair number of papers on dynamic node routing problems as it is presented in the survey of Pillac et al. (2013), dynamic arc routing problems have not received enough attention of the research community. This section summarises the works that deal with dynamic issues in arc routing problems.

2.2.1 Dynamic rural postman problem

Moreira (2007) introduce a dynamic RPP motivated by an industrial application on a high precision tools factory where pieces of different shapes have to be cut out of a surface by means of an electrified string. In a first phase, the pieces are nested in the given surface. In a second phase, an optimal cutting path is desirable for cutting out of pieces. The cutting surface is

suspended, which involves the falling of the surface portions cut out in succession. Any movement of the cutting tool generates an effective cut, unless of course it is performed over a region already cut out. Therefore the graph in which the cutting path is determined changes in a dynamic fashion along the cutting process itself. There are two considerations: every piece that is not completely cut out must not be traversed in its interior and there may not remain uncut pieces in areas that have been cut out. The problem is modeled as a rural postman problem where the cutting cost is minimized. Two heuristics are proposed to solve the problem. The first, called "higher up vertex" chooses the higher up vertex among the candidates reachable by uncut edges. The second, called "minimum empty path" chooses the nearest candidate as next vertex among the visible candidates. The two heuristics were tested on 10 real industrial instances containing between 19 and 52 pieces, and between 183 and 639 vertices. The proposed heuristics showed improvements of about 3% with respect to actual solutions.

2.2.2 Dynamic CARP

Dynamic issues in the CARP have been addressed in a few works. Handa et al. (2005) present an application in salt spreading on a road network in south Gloucestershire, England. The routing requirements on a particular day are linked to r*oad weather information system* (RWIS), which predicts road surfaces temperature and conditions across the road network within 20 minutes intervals. The information of road requiring treatment and the amount of salt needed for each road may change for each shift and even during a working shift. The problem is modeled as a classical CARP. A prototype system is proposed and consists of the RWIS and an evolutionary salting route optimization module. The optimization module has a memetic algorithm as optimization method. The algorithm uses the crossover EAX operator (Nagata, 1997) due to its search ability and a repair operator for offspring individuals is incorporated because the search space may include solutions that exceed the vehicle capacities. The prototype was examined on two typical shifts (one with 385 required edges and 11 vehicles and other with 97 required edges and 3 vehicles). A CARP is solved at the beginning of each shift with the predicted data. The results showed the effectiveness of the proposed method.

Yazici et al. (2014) present an interesting application for the *multi-robot sensor-based coverage problem*. A path planning must be designed such as every point in a given workspace is covered at least once by one of the robot's sensor. Initially, the robots are assumed to be at the

same starting point with equal energy capacities, but due to partially unknown nature of the workspace, the robots may face blockage on routes, and a fast re-planning is necessary considering remaining capacities and current positions of the robots. The problem is modeled as a CARP and the new plan is obtained by a modified Ulusoy's partitioning algorithm (1985). The algorithm was tested on two different real environments: on an indoor laboratory with 20 vertices and 2 robots, and on a larger laboratory with 90 vertices and up to 10 robots. The authors determined the maximum number of robots to be assigned to a given coverage task with efficient coverage cost.

Tagmouti et al. (2011) study the dynamic CARP *with time-dependent service cost*. This work is an extension of their previous works (Tagmouti et al., 2007, 2010) where the same problem is considered but on the static case. In the dynamic case the time interval where the service cost is minimal changes due to weather report updates, and therefore real-time modifications are required to the current routes. An adaptation of the VND of their previous work is presented as solution method. A starting solution is first computed with VND using service time cost functions based on an initial forecast. A simulated storm goes through the network and move along the x-axis. At different times, weather reports are received and update the storm speed. The VND is applied on a new static problem each time a weather report is received. In each static problem the graph is updated taking into account the already visited arcs. The algorithm was tested on a set of 60 generated instances with weather reports received every 5 minutes. The VND showed to be fast and allows the system to quickly use the new solution.

Weise et al. (2012) present a developmental solution method to the CARP that is suitable for dynamic scenarios. The method is based on genetic programming; it works with a solution space defined by all possible tours represented as a permutation of a subset of the required edges but a search space different to the solution space. The approach iteratively adds edges to a solution based on an environment's state. The main objective is to minimize the total traverse cost without consider limit on the number of vehicles. The solution method was tested on a large set of CARP benchmark instances for the static case. Some scenarios were derived for the dynamic case from 2 instances with 25 and 66 required edges by removing a certain number of edges and solving the problem with the new set of requirements.

Recently Liu et al. (2014) presented a memetic algorithm for the dynamic CARP. The algorithm is capable to solve CARP with variations in vehicle availability, road accessibility, new added task, canceled task, variation of traffic conditions, and variation of demands. The memetic algorithm incorporates a new split scheme with a path repair operator in order to handle the attributes of the problem. It combines features from global and local search, and has four key steps: split method, parent selection, crossover, and local search. 4 dynamic problems taken from Min et al. (2014) were used to test the algorithm including a case of 10 nodes, and 3 cases of 100 nodes with up to 33 tasks initially. The working of the proposed algorithm is illustrated using the 10-node example. The algorithm showed great potential for solving realistic dynamic CARP problems.

CHAPTER 3 ARTICLE 1 : THE RURAL POSTMAN PROBLEM WITH TIME WINDOWS

Marcela. Monroy-Licht[1,2] , Ciro Alberto Amaya[3], André Langevin[1,2]

[1] Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal, Canada
[2]Centre de recherche sur les réseaux d'entreprises, la logistique et le transport (CIRRELT), Montréal, Canada
[3]Departamento de Ingeniería Industrial, Universidad de Los Andes, Bogotá, Colombia

Abstract

The *rural postman problem with time windows* for the undirected case is introduced. The problem occurs in the monitoring of roads for black-ice detection. Different formulations are proposed and tested on sets of instances adapted from the literature. A cutting plane algorithm based on valid inequalities for the *traveling salesman problem* (TSP) *with time windows* and the *precedence constrained* TSP is presented as a solution method and tested on a set of real-life networks. Computational results show that this approach is able to solve to optimality instances with up to 104 required edges. At the end of the article the formulations for the undirected case are extended to the directed case.

Keywords: Rural postman problem, time windows, cutting plane algorithms, monitoring of roads

3.1 Introduction

The *rural postman problem with time windows* (RPPTW) involves finding a minimum- cost tour that goes through a set of required edges in a network. A vehicle leaves the depot, visits the required edges, and returns to the depot. A release time and a due time are given for each required edge. The tour is feasible if the visits are carried out during the defined time windows. Waiting times are allowed, i.e., the vehicle may arrive at any required edge earlier than its release time, but the service cannot start until the time window "opens". Costs of service are associated

with required edges and traversal costs with non-required edges. The vehicle may go through a required edge more than once, and the cost is normally lower when it does not service the edge.

The real-world application underlying this study is the monitoring of roads for black-ice detection (black-ice is a thin coating of clear or transparent ice on the pavement which is difficult for the drivers to see). This activity is carried out by the Ministry of Transport in the province of Quebec from mid-October to mid-December. The goal is to check the state of roads and take measures to prevent accidents. During this period, black ice on roads is almost invisible to the users; timely detection avoids pedestrian falls or automobile accidents.

Currently, a patrol must cover a network and generate reports about the state of the roads. The available information refers to a short-term weather forecast and a characterization of roads with high likelihood of black-ice formation; for example it is known that bridges and roads located near rivers are particularly susceptible to ice formation under certain meteorological conditions. The road segments to be checked are located in areas where the weather forecast indicates low temperatures and rain. The weather forecast induces the time windows for monitoring some of the road segments over a large region. Normally the patrol has enough time to visit all the roads defined previously in the schedule.

The RPPTW reduces to the *rural postman problem* (RPP) when the time-window constraints are not taken into account, so it is NP-hard. Little attention has been paid to the RPPTW. To the best of our knowledge the works of Nobert and Picard (1994) and Kang and Han (1998) are the only two related to the non-capacitated case and the one of Mullaseril et al. (1997) presents the capacitated version.

Nobert and Picard (1994) introduce a heuristic algorithm for the RPPTW. In their problem the required arcs are of two types: arcs that must be visited during the morning and arcs that may be visited all day long. They propose a heuristic method based on the solution of two rural path problems and on the computation of appropriate penalties. Numerical results are not published. Kang and Han (1998) consider the problem as a multiobjective optimization problem because they allow arrival at the required arcs after the due times, which incurs a cost penalty. The objective is to reduce the total traveling cost and total penalty. The authors present a genetic algorithm and compare three crossover operators.

Mullaseril et al. (1997) describe a feed distribution problem encountered on a cattle ranch in Arizona. The problem is cast as a collection of *capacitated rural postman problems with split-deliveries and time windows*. They present some heuristics and compare them with the working practices on the cattle ranch.

Another related problem, the RPP *with deadline classes*, has been studied by Letchford and Eglese (1998). They consider a single-vehicle arc routing problem in which the required edges are partitioned into a number of classes according to priorities, each class having its own deadline. An optimization algorithm is presented based on the use of valid inequalities as cutting planes. They tested the algorithm on a set of instances for the RPP from Corberán and Sanchis (1994) and found optimal solutions for all cases up to 67 required edges.

Our work addresses the undirected version of the problem. We assume that the costs of service are equal to the traversal costs, but our approaches could be easily modified if this is not the case. Our main contribution is to present the problem for a real-life application and several ways to model it. Three formulations are presented: one where the decision variables explicitly express the number of times an edge is traversed and two based on graphs equivalent to the original one. We then explore the third formulation, obtained when we transform the original problem to a *traveling salesman problem with time windows* and side constraints.

For the *traveling salesman problem* (TSP), polyhedral approaches have been extremely successful (Fischetti and Toth, 1997; Jünger, et al., 1995; Padberg and Rinaldi, 1991). Ascheuer et al. (2001) solve the *asymmetric* TSP *with time windows (*ATSP-TW) by a branch-and-cut method; they solve in a satisfactory way real-world instances of the control of a stacker crane in a warehouse. We have chosen to use the polyhedral approach. The RPPTW is formulated as an integer linear program that is solved by a cutting plane algorithm.

The paper is organized as follows. Section 3.2 presents three different models for the undirected case. In Section 3.3 we summarize the valid inequalities that we use as cutting planes in our algorithm. We briefly describe the solution algorithm in Section 3.4. Section 3.5 outlines the computational experiments. In Section 3.6 an extension of the formulations is given for the directed version of the problem, and Section 3.7 provides concluding remarks.

3.2 Undirected RPPTW

Let $G(V, E)$ be an undirected graph, where V is the set of vertices and E is the set of edges. Given a subset $E_R \subset E$ of required edges to service, the problem of finding a minimum cost tour traversing at least once all the required edges is known as the RPP.

In the RPP each edge $e \subset E_R$ is serviced exactly once, but can be traversed an additional number of times in a deadheading mode. Christofides et al. (1981) and Corberán and Sanchis (1991) present two formulations that use decision variables $x_e =$ number of times edge e is replicated in the optimal RPP solution if $e \in E_R$, and $x_e =$ number of times edge e is traversed if $e \in E \setminus E_R$. Although x_e can be bounded above by 1 if $e \in E_R$, and by 2 if $e \in E \setminus E_R$ (Eiselt, Gendreau, and Laporte, 1995), when time windows are considered this results is not valid. Indeed an edge could be traversed $|E_R| + 1$ times in the worst case (Gueguen, 1999).

An extension of these RPP formulations to RPPTW is difficult because it is not possible to associate a unique starting and completion time for an edge $e \subset E_R$. Arc routing problems with time windows are very hard to model directly without an extensive graph modification (Dror et al., 1997; Mullaseril, 1997; Mullaseril and Dror, 1996).

We turn on alternative modeling approaches. Some arc routing problems formulations use binary decision variables connecting edges (x_{ij}= 1 if edge j is traversed after edge i) or nodes (x_{ij}= 1 if node j is traversed after node i). We propose three formulations for the problem. The first considers a formulation based on edge linking variables. The other two formulations are based on transformed graphs and the decision variables linking nodes. The first transformation considers the required edges as nodes and joins them by means of arcs that represent the shortest paths among them in the original graph. The second transformation considers the nodes incident to the required edges and connects them with an arc that again corresponds to their shortest paths in the original graph.

3.2.1 Model on the edges

This formulation is based on the work presented by Gueguen (1999). He proposes a *mixed integer program* (MIP) formulation for the *capacitated arc routing problem with time windows* (CARPTW). Apparently this is the only formulation on edges for the CARPTW; however, the

author does not present numerical results. We modify Gueguen's formulation by adding a duplicate of each required edge to keep track of the direction in which a vehicle must travel along these edges. This is necessary to guarantee conservation of flow on the nodes of the network.

Consider the graph G and the subset $E_R \subset E$ previously defined. Let A be the set of edges that includes E, a duplicate $i°$ of each required edge $i \in E_R$, and an artificial edge "e_0" that represents the depot. The duplicate edges have the same cost and time windows as the originals. Let P be the set of pairs of edges $\{i, i°\}$ such that $i, i°$ correspond to the same required edge (the order $i, i°$ is determined arbitrarily), and R the set that contains all $i \in E_R$ and their duplicates. Additionally, $\dot{c}_i$ is the traversal cost of edge i, $\dot{T}_i$ is the traversal time of edge i, $[\dot{a}_i \ \ \dot{b}_i]$ is the time window for edge i; and for the edge "e_0" we set $\dot{c}_{e_0} = 0, \dot{T}_{e_0} = 0$, and $\dot{a}_{e_0} = 0$. M is a large integer number, which could be bound by $M = \max_{i \in E} \dot{T}_i \ |E||E_R| + 1$, and δ_i is the set of vertices incident to edge i. $m = |E_R| + 1$ is the maximum number of times an edge can be traversed. The decision variables defined hereafter allow us to keep track of the number of times (each time corresponds to a "copy" of an edge) that the vehicle traverses each edge.

Let the decision variables $x_{ijkl} = 1$ if copy l of edge j is traversed immediately after copy k of edge i, and 0 otherwise; and let t_{ik} be the time to start traversing copy k of edge i. The formulation on the edges is as follows:

$$\text{minimize} \sum_{i \in A} \sum_{\substack{j \in A| \\ \delta_j \cap \delta_i \neq \emptyset, j \neq i}} \sum_{k=1}^{m} \sum_{l=1}^{m} \dot{c}_i \, x_{ijkl} \tag{3.1}$$

s.t.:

$$\sum_{\substack{j \in A| \\ \delta_j \cap \delta_i \neq \emptyset, j \neq i}} \sum_{l=1}^{m} x_{ij1l} + \sum_{\substack{j \in A| \\ \delta_j \cap \delta_{i°} \neq \emptyset, j \neq i°}} \sum_{l=1}^{m} x_{i°j1l} \ = 1 \ \forall \{i, i°\} \in P \tag{3.2}$$

$$\sum_{\substack{j \in A| \\ \delta_j \cap \delta_{e_0} \neq \emptyset, j \neq e_0}} \sum_{l=1}^{m} x_{e_0 j1l} \geq 1 \tag{3.3}$$

$$\sum_{\substack{i\in A|\\ \delta_i\cap\delta_j\neq\emptyset\,,i\neq j}}\sum_{k=1}^{m} x_{ijkl} = \sum_{\substack{i\in A|\\ \delta_i\cap\delta_j\neq\emptyset\,,i\neq j}}\sum_{k=1}^{m} x_{jilk} \quad \forall\, j\ \in A\,, l=1,\dots,m \tag{3.4}$$

$$\sum_{\substack{j\in A|\\ \delta_j\cap\delta_i\neq\emptyset\,,i\neq j,j\neq i^\circ}}\sum_{k=1}^{m}\sum_{l=1}^{m} x_{ijkl} + \sum_{\substack{j\in A|\\ \delta_j\cap\delta_{i^\circ}\neq\emptyset\ j\neq,i^\circ,j\neq i}}\sum_{k=1}^{m}\sum_{l=1}^{m} x_{i^\circ jkl} \geq 1\ \forall\{i,i^\circ\}\in P \tag{3.5}$$

$$t_{ik}+\dot{T}_i \leq t_{jl} + M\left(1-x_{ijkl}\right) \quad \forall\, i\ \in A,\ j\in A|\ \delta_i\cap\delta_j\neq\emptyset\,, i\neq j, j\neq e_0$$

$$k,l=1,\dots,m \tag{3.6}$$

$$t_{i1}\ \geq \dot{a}_i\ \forall\, i\ \in R\ \cup\{e_0\} \tag{3.7}$$

$$t_{i1}\leq \dot{b}_i\ \forall\, i\ \in R \tag{3.8}$$

$$\sum_{l=1}^{m} x_{ijkl}\leq 1 \quad \forall\, i\ \in A, j\in A|\delta_i^{(-)}=\delta_j{}^{(+)}\,, i\neq j,\ \ k=1,\dots m \tag{3.9}$$

$$x_{ijkl}\in\{0,1\}\ \ \forall\, i\ \in A, j\in A|\delta_i\cap\delta_j\neq\emptyset\,, i\neq j, \qquad k,l=1,\dots,m \tag{3.10}$$

$$t_{ik}\in\mathbb{R}^+\ \forall\ i\ \in\ A\,, \qquad k=1,\dots m \tag{3.11}$$

The objective is to minimize the total traversal cost. Services are ensured by constraints (3.2). Constraint (3.3) forces the tour to start at the depot. Flow conservation is ensured by constraints (3.4). Constraints (3.5) avoid solutions with subcycles between any pair of edges in P that represents the same edge in A. Inequalities (3.6), (3.7), and (3.8) are the time-window constraints. Constraints (3.9), not in Gueguen (1999), allow us to reduce the number of equivalent solutions. Finally, constraints (3.10) and (3.11) define the decision variables.

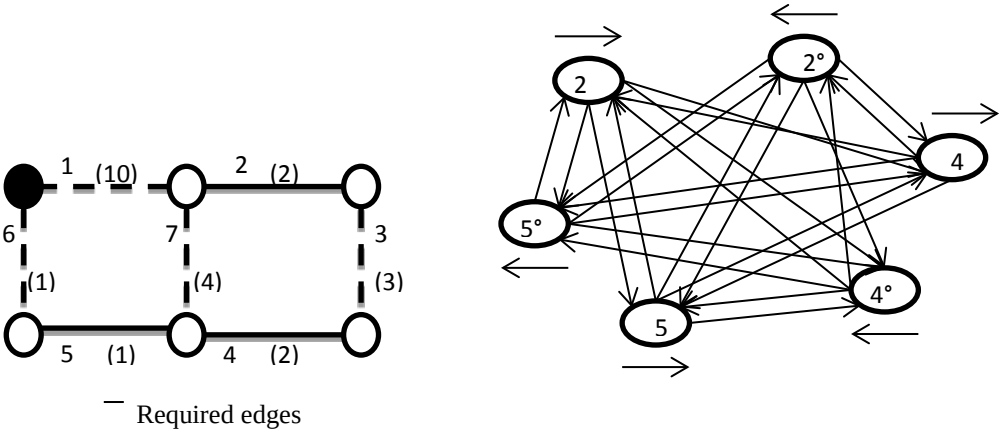
In this model, the original graph is extended firstly by making a duplicate of all required edges, secondly by making $|E_R|+1$ copies of all edges. If the number of required edges is large, the formulation is intractable. The model also uses a large real value M that generates weak relaxations and numerical difficulties in the solution methods.

3.2.2 Model on the required edges

Since the model on the edges (Section 3.2.1) is not practical, we propose an equivalent formulation. On the graph $G = (V, E)$ two required edges can be connected successively on a route in four ways, depending on the traversal direction of the two edges. This leads us to define the problem on a new graph $G_0 = (N, A_0)$.

For each required edge i in G we define two nodes i, i° in G_0, where $i, i^{\circ} \in N$ represent the two possible directions in which edge i in G can be traversed. Each arc of A_0 connects a node $i \in N$ with a node $j \in N$ if they do not correspond to the same edge in G. The cost of the arc that connects node i to node j in G_0 is equal to the cost of the edge represented by i plus the length of the shortest path in G from the final node of the edge represented by i to the initial node of the edge represented by j, according to the traversal directions.

Figure 3.1 shows an example of the transformation. The original graph is presented in a). The depot is located at the black node; the edge indices are shown near each edge; and the traversal costs are in parentheses. There are three required edges (2, 4, and 5). The transformed graph is illustrated in b). Its nodes are labeled with the same number as their corresponding required edges. Arrows over and under the nodes indicate the traversal direction that each node represents. Note that pairs of nodes corresponding to the same required edge are not connected.



a) Original graph b) Transformed graph

Figure 3.1: Transformed graph for the model on the required edges

Finally we add an artificial node labeled "0" to represent the depot. We join "0" to all nodes in N by means of two arcs with a cost equal to the length of the shortest path in G from the depot to the edges, and from the edges to the depot respectively, again according to the traversal direction. Table 3.1 indicates the cost of the arcs of the transformed graph. The time window for each node is the same as for the corresponding edge.

Table 3.1: Costs of the transformed graph – Model on the required edges

From	To						
	0	**2**	**2°**	**4**	**4°**	**5**	**5°**
0	0	6	7	2	4	1	2
2	9	0	0	7	5	8	7
2°	8	0	0	6	7	7	6
4	6	7	5	0	0	5	4
4°	4	6	7	0	0	3	2
5	3	5	6	1	3	0	0

In the transformed graph G_0 we look for a minimum-cost tour that visits one of the two nodes that represent the same required edge. The tour starts and ends at the depot, and the visits must satisfy the time windows.

Let us consider the set C that includes the pairs of nodes $\{i, i°\}$, where $i, i° \in N$, such that $i, i°$ represent the same required edge. We consider the following parameters: c_{ij} is the traversal cost from node i to node j, T_{ij} is the traversal time from node i to node j, $[a_i\ b_i]$ is the time window for node i, and $M_{ij} = \max\left\{ b_i + T_{ij} - a_j ,0\right\}$. We set $a_0 = 0$ for the depot node.

We define the decision variables x_{ij} to be equal to 1 if node j is serviced immediately after node i, and 0 otherwise, and t_i to be the arrival time at node i. The formulation on the required edges is as follows:

$$\text{minimize} \sum_{i \in N \cup \{0\}} \sum_{\substack{j \in N \cup \{0\} | j \neq i, \\ \{i,j\} \notin C}} c_{ij} x_{ij} \qquad (3.12)$$

s.t.:

$$\sum_{\substack{j \in N \cup \{0\}| \\ j \neq i, i^{\circ}}} (x_{ij} + x_{i^{\circ}j}) = 1 \quad \forall \ \{i, i^{\circ}\} \ \in C \tag{3.13}$$

$$\sum_{\substack{i \in N \cup \{0\}| \\ i \neq j, j^{\circ}}} (x_{ij} + x_{ij^{\circ}}) = 1 \quad \forall \ \{j, j^{\circ}\} \ \in C \tag{3.14}$$

$$\sum_{j \in N} x_{0j} = 1 \tag{3.15}$$

$$\sum_{j \in N} x_{j0} = 1 \tag{3.16}$$

$$t_i + T_{ij} \leq t_j + M_{ij}\left(1 - x_{ij}\right) \ \forall \ i \ \in N \ , j \ \in N \cup \{0\}| \ \ i \ \neq j, \ \ \{i, j\} \notin C \tag{3.17}$$

$$a_i \leq t_i \ \forall \ i \ \in N \cup \{0\} \tag{3.18}$$

$$t_i \leq b_i \ \forall \ i \ \in N \tag{3.19}$$

$$x_{ij} \leq \sum_{\substack{k \in N \cup \{0\}| \\ k \neq j}} x_{jk} \ \ \forall \ i \ \in N \cup \{0\}, j \in N \cup \{0\}| \ i \neq j, \ \ \{i, j\} \notin C \tag{3.20}$$

$$x_{ij} \in \ \{0,1\} \ \forall \ i \ \in N \cup \{0\}, j \ \in N \cup \{0\} | i \neq j, \ \ \{i, j\} \notin C \tag{3.21}$$

$$t_i \in \mathbb{R}^{+} \ \forall \ i \ \in N \cup \{0\} \tag{3.22}$$

The objective is to minimize the total traversal cost. Constraints (3.13) and (3.14) ensure that only one node is included in the tour for each pair of nodes that represent the same required edge. Constraints (3.15) and (3.16) force the tour to start and end at the depot. The time-window constraints are (3.17), (3.18), and (3.19). Constraints (3.20) guarantee flow conservation. Finally, the decision variables are defined in (3.21) and (3.22).

3.2.3 Model on the nodes

We now propose a transformation from the original problem to an equivalent problem on nodes. We define a new graph $G_1 = (N_1, A_1)$, where all the vertices incident to the required edges

in the original graph $G = (V, E)$ are included in the set of nodes N_1. It should be pointed out that if a vertex of V is incident to more than one required edge in G, then this vertex will have as many copies in N_1 as the number of incident required edges in G. A_1 is the set of arcs that connects the nodes of N_1. The cost of an arc that joins node i to node j is equal to the cost of the edge that starts at node i plus the length of the shortest path in G from the final vertex of that edge to the initial vertex j of the other edge. We set the cost to zero when nodes i and j represent vertices incident to the same edge in G or when $i = j$. We obtain a complete directed graph. We illustrate on an example the original graph in Figure 3.1a and the transformed graph in Figure 3.2 Note that there are two nodes labeled "4" and "4°" because they represent vertex "4" of Figure 3.1a, which is incident to two required edges.

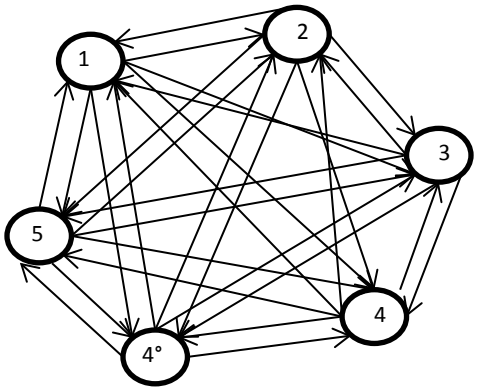


Figure 3.2: Transformed graph for the model on the nodes

We add an artificial node "0" to represent the depot. We join "0" to all nodes in N_1 by means of two arcs with costs equal to the length of the shortest path in G from the depot to the vertices, and from the vertices to the depot respectively. The time windows of each required edge in G are assigned to its incident nodes.

In the transformed graph, we look for a minimum-cost tour that starts and ends at the depot and satisfies the time windows for all nodes. Additionally, two nodes incident to the same required edge must be placed one after the other in the tour sequence. Table 3.2 shows the cost matrix for the transformed graph.

Table 3.2: Costs of the transformed graph – Model on the nodes

From	To						
	0	**1**	**2**	**4**	**3**	**5**	**4°**
0	0	6	7	2	4	1	2
1	9	0	0	7	5	8	7
2	8	0	0	6	7	7	6
4	6	7	5	0	0	5	4
3	4	6	7	0	0	3	2
5	3	5	6	1	3	0	0

Let us define C_0 as the set of pairs of nodes that are incident to the same required edge, and U as the set of nodes that includes one of the two nodes incident to the same required edge. Furthermore, let the parameters c_{ij}, T_{ij}, $[a_i\ b_i]$, a_0, and M_{ij} and the decision variables x_{ij} and t_i be as defined in Section 2.2. The formulation on the nodes is as follows:

$$\text{minimize} \sum_{i \in N_1 \cup \{0\}} \sum_{j \in N_1 \cup \{0\} | j \neq i} c_{ij} x_{ij} \tag{3.23}$$

s.t.:

$$x_{ii^\circ} + x_{i^\circ i} = 1 \quad \forall\ \{i, i^\circ\} \in C_0 \tag{3.24}$$

$$\sum_{\substack{j \in N_1 \cup \{0\} | \\ j \neq i}} x_{ij} = 1 \quad \forall\ i \in N_1 \cup \{0\} \tag{3.25}$$

$$\sum_{\substack{i \in N_1 \cup \{0\} | \\ i \neq j}} x_{ij} = 1 \quad \forall\ j\ \in N_1 \cup \{0\} \tag{3.26}$$

$$t_i + T_{ij} \leq t_j + M_{ij}(1 - x_{ij})\ \ \forall\ i\ \in N_1\ ,\ j\ \in N_1 \cup \{0\} |\ \ i\ \neq j \tag{3.27}$$

$$a_i \leq t_i\ \forall\ i\ \in U\ \cup \{0\} \tag{3.28}$$

$$t_i \leq b_i\ \forall\ i\ \in U \tag{3.29}$$

$$t_i = t_{i^\circ}\ \forall\ \{i, i^\circ\} \in C_0 \tag{3.30}$$

$$x_{ij} \in \{0,1\} \ \forall \ i \ \in N_1 \cup \{0\}, j \ \in N_1 \cup \{0\} | \ i \neq j \tag{3.31}$$

$$t_i \in \mathbb{R}^+ \ \forall \ i \ \in N_1 \cup \{0\} \tag{3.32}$$

The objective is to minimize the total traversal cost. Constraints (3.24) are related to the required services. Constraints (3.25) and (3.26) ensure that each node is visited. The time-window constraints are (3.27), (3.28), (3.29), and (3.30). The decision variables are defined in (3.31) and (3.32).

For the model on the edges we cannot associate a unique starting and completion time with each edge; we need the indices *k* and *l* to identify the number of times each edge would be traversed in a deadheading mode for the minimum distance objective. Therefore we must augment the initial graph $k \times l$ times. The graph for models based on transformations includes only the incident nodes to required edges or the required edges. The other edges (not required) are considered only for getting the shortest path among required edges. Consequently, the variables do not need an extra index to identify the number of times edges are traversed in a deadheading mode (required edges are visited no more than once in the modified graph).

The models based on the transformations have fewer variables and constraints than the model on the edges. They also use large integer values M_{ij}, but these can be bounded to minimum values that allow us to find feasible solutions, therefore these formulations are better bounded.

3.3 Valid inequalities

We focus on the model on the nodes (Section 3.2.3), taking advantage of its structure. This model has elements of the *precedence constrained asymmetric* TSP (PC-ATSP). Polyhedral approaches to solve problem instances to optimality are known to work well for the PC-ATSP (Ascheuer et al. 2000), as already mentioned, for the TSP. We study some of the known valid inequalities with respect to the two problems that are also valid for the formulation (3.23)–(3.32). In the following, we summarize the classes of inequalities that we use in our solution method.

Notation

Given the set of arcs A_f, for any arc set $W \subseteq A_f$ we define $x(W) \coloneqq \sum(x_{ij} \,|\{i,j\} \in W)$. Given the set of nodes V_f that includes the depot "0", for any two node sets $S, T \subseteq V_f$ we define $(S{:}T) \coloneqq \left\{\{i,j\} \in A_f \mid i \in S, j \in T\right\}$ and write $x(S{:}T)$ for $x\big((S{:}T)\big)$.

Lifted t-bounds. Desrochers and Laporte (1991) observed that the bounds of the t-variables (see inequalities 3.28 and 3.29) can be strengthened. Indeed, let $a_{ji} = \max\{0, a_j - a_i + T_{ji}\}$ and $b_{ij} = \max\{0, b_i - b_j + T_{ij}\}$. Then the inequalities

$$a_i + \sum_{\substack{j=1 \\ |j \neq i}}^{n} a_{ji} x_{ji} \leq t_i \;\; \forall\, i \in V_f \setminus \{0\} \tag{3.33}$$

$$b_i - \sum_{\substack{j=1 \\ |j \neq i}}^{n} b_{ij} x_{ij} \geq t_i \;\; \forall\, i \in V_f \setminus \{0\} \tag{3.34}$$

are valid for the formulation (3.23)–(3.32).

Strengthened MTZ-inequalities. Desrochers and Laporte (1991) propose a lifted version of the MTZ subtour-elimination constraints (3.27). Let $\bar{a}_{ji} = \max\{T_{ji},\; a_i - b_j\}$ and $M_{ij} \geq b_i + T_{ij} - a_j$. Then for all $i, j = 1, \ldots, n, i \neq j$ the inequality

$$t_i + T_{ij} - \big(1 - x_{ij}\big)M_{ij} + \big(M_{ij} - T_{ij} - \bar{a}_{ji}\big)\, x_{ji} \;\leq t_j \tag{3.35}$$

is valid for the formulation (3.23)–(3.32).

According to Desrochers and Laporte (1991), when precedence relations exist, the MTZ-inequalities can be further strengthened. Assume $i \prec j$. Since i must be scheduled before j, we have $t_i \leq t_j$, and the inequality

$$t_i + T_{ij} x_{ij} \leq t_j \tag{3.36}$$

is also valid. If $b_i + T_{ij} \leq a_j$ holds, inequality (3.36) can be strengthened to

$$t_i + T_{ij} x_{ij} \leq a_j \tag{3.37}$$

Subtour elimination constraints. We include the subtour elimination constraints, since they are the best known inequalities for the Asymmetric Traveling Salesman polytope (Balas et al. 1995). These inequalities $x(S:S) \leq |S| - 1$ can be written in the equivalent cut form

$$x(S : \bar{S}) \geq 1 \ \forall \ S \neq \emptyset, S \subset V_f \tag{3.38}$$

where $\bar{S} \coloneqq V_f \setminus S$, and (3.6) is valid for the formulation (3.23)–(3.32).

The Predecessor/Successor inequalities. The PC-ATSP is a relaxation of the ATSP-TW. We use some valid inequalities for the PC-ATSP that allow us to strengthen the subtour elimination inequalities (3.38). Balas et al.(1995) introduced these classes of inequalities.

For $S \subseteq V_f \setminus \{0\}$, $\bar{S} \coloneqq V_f \setminus S$, the predecessor inequality (π-inequality)

$$x\big(S \setminus \pi(S) : \bar{S} \setminus \pi(S)\big) \geq 1 \tag{3.39}$$

and the successor inequality (σ-inequality)

$$x\big(\bar{S} \setminus \sigma(S) : S \setminus \sigma(S)\big) \geq 1 \tag{3.40}$$

are valid for the formulation (3.23)–(3.32).

For any given $i, k \in V_f \setminus \{0\}$ such that $\pi(i) \neq \emptyset$, $\sigma(k) \neq \emptyset$, and any $S \subset V_f$ such that $i, k \in S$, the inequalities

$$x\big(S \setminus \pi(i) : \bar{S} \setminus \pi(i)\big) \geq 1 \tag{3.41}$$

$$x\big(\bar{S} \setminus \sigma(k) : S \setminus \sigma(k)\big) \geq 1 \tag{3.42}$$

are called weak π- and weak σ-inequalities respectively.

3.4 Solution algorithm

We implement the following algorithm to solve the Undirected RPPTW.

3.4.1 Data preprocessing

Data preprocessing is important for efficient implementations. It allows the construction of tighter equivalent formulations of the problems, such that no optimal solution of the original problem is lost and each solution of the tighter problem corresponds to a solution of the original problem.

The structures of the formulations on the required edges (Section 3.2.2) and on the nodes (Section 3.2.3) permit such a preprocessing procedure. It is based on the work of Ascheuer et al. (1999). We tighten the time windows iteratively until no more changes are made. We then

identify precedence relations, fix variables permanently, and detect infeasible paths of size two and three to reduce the set of variables.

We now present the separation procedures for the classes of valid inequalities (3.38)–(3.42).

3.4.2 Cutting plane algorithm

Initial linear program. We solve the relaxation of the model on nodes (3.23)–(3.32), i.e., when the decision variables x_{ij} are restricted to be nonnegative and less than or equal to one. Constraints (3.33) and (3.34) are included in the initial model instead of constraints (3.28) and (3.29) because the former are stronger. We also include the strengthened MTZ-inequalities (3.35), (3.36), and (3.37) instead of the MTZ-inequalities (3.38) when possible.

Separation routines. Let (x^*, t^*) be a solution where x^* is fractional. We want to identify a member of a family F of valid inequalities listed in Section 3.3 for the formulation on the nodes that is violated by x^* or else show that x^* satisfies all members of *F*. The implemented separation procedures are an adaptation of routines described in the literature.

- Subtour elimination constraints: For the cutset inequalities (3.38) we can solve the separation problem by computing the connected component T that includes the depot in the graph G^* induced by $x_{ij}{}^* > 0$. If this component does not include all the nodes in V_f, the subtour elimination constraint is violated by x^*, and we obtain the set S that includes all nodes in T. This procedure detects inequalities (3.38) which are violated only in the case where there is no path in G^* from the depot to any $j \in V_f \setminus \{0\}$.
- Predecessor inequalities: We implement the exact separation algorithm presented by Balas et al.(1995) for the separation problem of predecessor inequalities. Although this algorithm detects only a violated weak π-inequality, if one exists, rather than a stronger π-inequality of the class (3.39), the detected violated inequality (3.41) can be replaced with a strictly stronger violated inequality of the class (3.39) when we include $\pi(S)$ instead of $\pi(j)$. If we apply the algorithm for $j \in V_f \setminus \{0\}$ such that $\pi(j) = \emptyset$, we detect the known cutset inequality, and

obtain an algorithm that simultaneously solves the separation problem for both the subtour elimination inequalities and the π-inequalities.

- The successor inequalities: With a similar procedure to Balas et al.(1995) we can detect if x^* violates a weak σ-inequality. For any fixed j with $\sigma(j) \neq \emptyset$, delete $\sigma(j)$ from V_f and in the resulting network with arc capacities ${x_{ij}}^*$ try to send one unit of flow from node 0 to node j. If this is possible, all inequalities (3.42) associated with the given j are satisfied by x^*; otherwise the minimum capacity identified by the failed attempt to send a unit of flow specifies the σ-inequality most violated by x^*. We reverse the sets S and $\bar{S}$, i.e., S is replaced by $\bar{S}$ and vice versa. As in the previous case, if a violated inequality (3.42) is found, it is replaced with a strictly stronger violated inequality of the class (3.40), when we include $\sigma(S)$ instead of $\sigma(j)$.

Steps for the separation routine

- Subtour elimination constraint routine.
- "Shrinking": The separation algorithm for the predecessor/successor inequalities implies the computation of the maximum flow for each pair $i, j \in V_f$. We use "shrinking" procedures (Padberg and Rinaldi, 1990) to reduce the problem size and to avoid as many maximum-flow calculations as possible. "Shrinking" checks whether or not certain nodes lie on the same side of a minimum-capacity cut. If the results are positive, the subset is contracted or "shrunk" to a single node. We contract nodes i and j if they are incident to the same required edge. Also, we contract nodes i and j if ${x_{ij}}^* = 1$ in the fractional solution x^*.
- Predecessor inequalities: We use the separation problem for the predecessor inequalities, and we simultaneously check the subtour elimination constraints when there is one connected component in the fractional solution x^*.
- Successor inequalities: We use the separation problem for the successor inequalities, and we simultaneously check the subtour elimination constraints when there is one connected component in the fractional solution x^*.

We generate at most one cutting plane for each separation routine per iteration. The linear problems are solved using standard parameters of CPLEX 12.4.0.

3.4.3 Solution of the MIP program

We stop the cutting plane algorithm whenever the last 10 linear problems produce no improvement in the lower bound, or in case the improvement is less than 0.1%, or when the running time reaches three hours. In those scenarios the decision variables x_{ij} are restricted to be binary, we add the valid cuts, if any and we solve the problem using the callable library of CPLEX 12.4.0.0. with its default parameters except that the number of threads is set to 1.

3.5 Computational results

In this section we describe the results of the comparison of the models on a set of randomly generated instances and the performance of our cutting plane algorithm which was tested also on a set of instances based on the real network of the Estrie region in Quebec. Our implementation is coded in Python 2.6 and runs on a 2.38 GHz AMD 250.

3.5.1 Generated instances

There are no published benchmark instances for the undirected RPPTW. We modified the CARP-TW instances of Wøhlk (2005). The author combines five values for the number of nodes ($\{10,13,20,40,60\}$) and the number of edges ($\{15,23,31,69,90\}$) and generates different graphs for each combination.

We selected some required edges randomly and found a path through all of them using the nearest-neighbor heuristic. Then, we established the time-window intervals by reducing and extending by 10, 30, and 50% the values of the arrival time given by the heuristic. In this way we label the width of the time windows with $\{10,30,50\}$. When we combined the percentage of required edges $\{10,30,50\}$ and the width of the time windows $\{10,30,50\}$ we generated 225 instances. All the instances can be downloaded from http://ftpprof.uniandes.edu.co/~pylo/inst/RPPTW/instances.htm.

3.5.2 Instances based on the Estrie network

We tested the cutting plane algorithm on a real undirected network that represents a part of the Estrie administrative region in the province of Quebec, Canada. The network has 140 nodes and

187 edges. We simulated nine weather forecasts for one day with 4 or 5 time slots, each time slot defining the time windows. If any edge is located in a time slot with rain forecasted, the edge will be required to be visited in its respective time slot. Figure 3.3 shows an example of simulated weather forecasting for Estrie region for one time slot. The colors represent a different probability of rain. We define the cost of traversal as the length of the road multiplied by a fractional number in order to get a scalar representation, and the time of traversal proportional to this value.

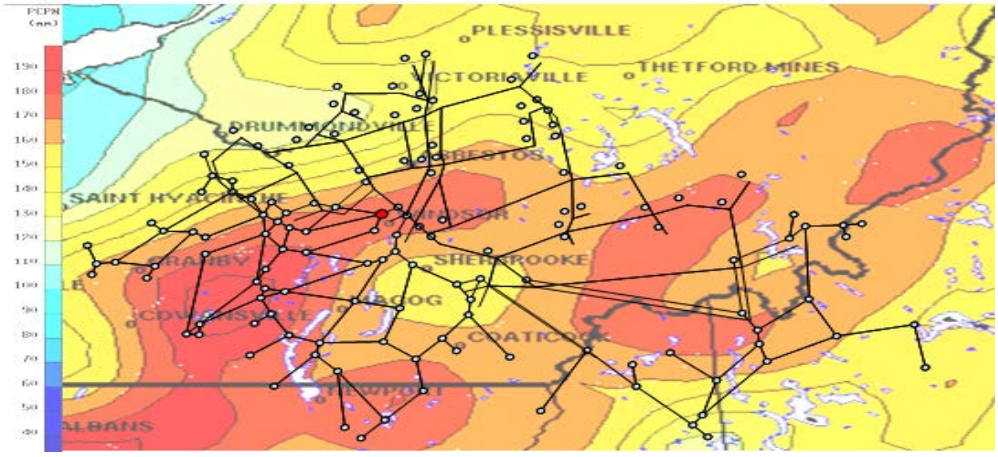


Figure 3.3: Estrie network – Weather forecast for one time slot

3.5.3 Preprocessing

As noted earlier, the structures of the formulations on the required edges (Section 3.2.2) and on the nodes (Section 3.2.3) allow data preprocessing. Table 3.3 shows the effect of data preprocessing, giving the average percentage of removed variables. In general, there was a considerable reduction in the problem size. When the time windows are tighter more variables can be fixed.

Table 3.3: Reduction in number of variables

T.W. Width	Model on the required edges (%)	Model on the nodes (%)
10	47.44	40.15
30	36.96	30.43
50	26.09	21.88

3.5.4 Tests

In this section we present a summary of the presented models performance as well the cutting plane algorithm performance. Detailed results for each instance are available at http://ftpprof.uniandes.edu.co/~pylo/inst/RPPTW/anex1-results.htm. We solved the models using CPLEX 12.4.0 with the callable library, setting a running time of up to three hours excluding the data-preprocessing time.

We compare only the models based on the transformations because the model on the edges (Section 3.2.1) is too large even when the number of copies of required edges is small. Table 3.4 summarizes the results for subsets of instances grouped by size. Column *TW* lists the different time-window widths, *n* is the number of nodes, *e* is the number of edges, $|R|$ is the average number of required edges, *N* is the total number of instances in the set, *sol* is the number of instances solved to optimality, and *t* is the average running time in seconds.

The model on the nodes (Section 3.2.2) is superior to the model on the required edges (Section 3.2.3) because it finds the optimal solution for more instances, the optimal solution for harder instances, and the computational times are smaller.

The results for the cutting plane algorithm are summarized in the next columns. Column *gap* shows the average gap for the set of instances between the lower bound given by the added cuts and the optimal solution value. Columns *mic* and *mac* show the minimum and maximum number of added cuts. Finally, *ave t* and *t max* present the average and maximum running times.

Our algorithm was able to solve 222 of 225 instances. 78 problems were solved to optimality at the root node of the search tree, using only cutting planes. We solved to optimality 10 of the

largest instances (60 nodes, 90 edges and 45 required edges); the average computational time for solving these 10 instances is 814.9 seconds.

Table 3.4: Models comparison and cutting plane algorithm

					Model on the required edges		**Model on the nodes**		**Cutting plane algorithm (model on the nodes)**					
TW	*n*	*e*	*|R|*	*N*	*sol*	*t*	*sol*	*t*	*sol*	*gap %*	*mic*	*mac*	*ave t*	*t max*
10				9	9	0.003	9	0.004	9	0	0	2	0.057	0.19
30	10	15	5	9	9	0.016	9	0.010	9	0	0	5	0.08	0.13
50				9	9	0.018	9	0.018	9	1.2	0	22	0.21	0.82
10				27	27	0.016	27	0.011	27	0.2	0	10	0.21	0.81
30	13	23	7.4	27	27	0.267	27	0.069	27	1.6	0	17	0.61	2.85
50				27	27	0.286	27	0.098	27	2.6	0	22	0.50	1.99
10				9	9	0.050	9	0.028	9	0.1	0	6	0.66	2.31
30	20	31	10	9	9	4.901	9	1.641	9	2.7	0	19	1.30	7.27
50				9	9	8.491	9	0.590	9	4.6	0	17	1.82	6.53
10				18	18	51.84	18	2.43	18	1.1	4	25	11.18	25.19
30	40	69	21	18	15	1085.69	17	824.08	18	3.5	2	25	17.11	83.02
50				18	14	285.88	17	510.49	17	5.2	0	24	764.52	6675.22
10				12	11	405.02	12	104.31	12	2.1	5	27	17.83	60.23
30	60	90	27	12	7	361.78	10	1398.49	11	1.2	0	25	168.83	1659.91
50				12	5	6028.02	8	2354.40	11	4.0	3	24	1101.85	6030.17

Table 3.5: Cutting plane on real instances

Instance	***T.W***	***|R|***	***O.F.***	***suc***	***pc***	***cc***	***gap***	***t***
Inst-00	Tight	74	--	--	--	--	--	--
Inst-01	Intermediate	74	259.7	2	10	11	7.6	165.46
Inst-02	Wide	74	--	--	--	--	--	--
Inst-03	Tight	104	289.2	2	11	11	2.3	202.29
Inst-04	Intermediate	104	--	--	--	--	--	--
Inst-05	Wide	104	--	--	--	--	--	--
Inst-06	Tight	93	299.7	2	11	10	7	193.41
Inst-07	Intermediate	93	299.7	2	11	9	7	208.75
Inst-08	Wide	93	299.7	1	11	11	7	137.76

The results of the cutting plane algorithm on the set of real instances are summarized in Table 3.5. Colum *O.F.* presents the value of the objective function, *suc* the number of cuts added for subtour elimination, *pc* the cuts for precedence relations and *sc* the cuts for successor relations. We were able to solve 5 of the 9 instances in less than 3.5 minutes. Instances that do not show values were not solved by the algorithm within 3 hours.

3.6 Directed case

In this section we present two formulations for the directed case, which are extensions of the formulations for the undirected problem. The first is a formulation on the arcs, i.e., the decision variables express the number of times an arc is traversed. The second is based on a transformed graph, where the required arcs are connected by an arc that represents the shortest path among them. We carry out tests to evaluate the performance of the models.

3.6.1 Model on the arcs

As in the undirected case, this formulation is based on the work presented by Gueguen (1999).

Consider a directed graph $G_d(V_d, D)$, where V_d is the set of vertices, $0 \in V_d$ represents the depot , and D is the set of arcs. Let $\hat{A}$ be the set that includes D and two artificial arcs "a_l" and "a_e" leaving and entering the depot respectively. $\hat{R}$ is the set of required arcs plus "a_l" and "a_e". Furthermore, we define the following parameters: $\hat{m} = |\hat{R}| + 1$ is the maximum number of times an arc can be traversed in a feasible solution, $\hat{c}_i$ is the traversal cost of arc i, $\hat{T}_i$ is the traversal time of arc i, $[\hat{a}_i\ \hat{b}_i]$ is the time window for arc i, M is a large real value which could be bounded by $M = \max_{i \in E} \dot{T}_i\ |D||\hat{R}| + 1$, $\delta_i^{(+)}$ is the end vertex of arc i, and $\delta_i^{(-)}$ is the initial vertex of arc i.

The decision variables for this formulation are defined as $x_{ijkl} = 1$ if copy l of arc j is traversed immediately after copy k of arc i, and 0 otherwise, and t_{ik} indicates the arrival time at copy k of arc i. The model is given below:

$$\text{minimize} \sum_{i \in \hat{A}} \sum_{\substack{j \in \hat{A} | \\ \delta_j^{(-)} = \delta_i^{(+)}}} \sum_{k=1}^{\hat{m}} \sum_{l=1}^{\hat{m}} \hat{c}_i\, x_{ijkl} \tag{3.43}$$

s.t.:

$$\sum_{\substack{j\in\hat{A}| \\ \delta_j^{(-)}=\delta_i^{(+)}}} \sum_{l=1}^{\hat{m}} x_{ij1l} \geq 1 \ \forall \ i \in \hat{R} \tag{3.44}$$

$$\sum_{\substack{i\in\hat{A}| \\ \delta_i^{(+)}=\delta_j^{(-)}}} \sum_{k=1}^{\hat{m}} x_{ijkl} = \sum_{\substack{i\in\hat{A}| \\ \delta_i^{(-)}=\delta_j^{(+)}}} \sum_{k=1}^{\hat{m}} x_{jilk} \quad \forall j \in \hat{A}, \quad l = 1, \dots, \hat{m} \tag{3.45}$$

$$t_{ik} + \hat{T}_i \leq t_{jl} + M\left(1 - x_{ijkl}\right) \ \forall \ i \in \hat{A}, j \in \hat{A} | \delta_i^{(-)} = \delta_j^{(+)}, \ \delta_i^{(+)} \neq \{a_e\}, \quad k, l = 1, \dots, \hat{m} \tag{3.46}$$

$$t_{i1} \geq \hat{a}_i \ \forall \ i \in \hat{R} \tag{3.47}$$

$$t_{i1} \leq \hat{b}_i \ \forall \ i \in \hat{R} \ | i \notin a_e \tag{3.48}$$

$$\sum_{l=1}^{\hat{m}} x_{ijkl} \leq 1 \quad \forall \ i \in \hat{A}, j \in \hat{A} | \delta_i^{(-)} = \delta_j^{(+)}, \ k = 1, \dots \hat{m} \tag{3.49}$$

$$x_{ijkl} \in \{0,1\} \ \forall \ i \in \hat{A}, j \in \hat{A} | \delta_i^{(-)} = \delta_j^{(+)}, k, l = 1, \dots, \hat{m} \tag{3.50}$$

$$t_{ik} \in \mathbb{R}^+ \ \forall \ i \in \hat{A}, \quad k = 1, \dots \hat{m} \tag{3.51}$$

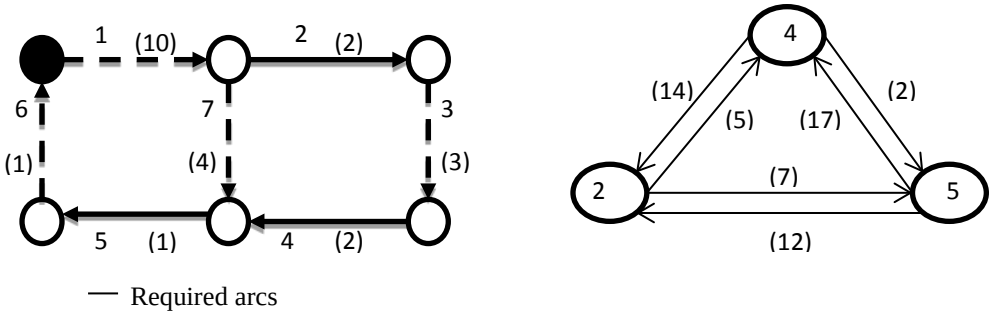
The objective function minimizes the total traversal cost. Services are ensured by constraints (3.44). Constraints (3.45) define the flow conservation. Inequalities (3.46), (3.47), and (3.48) are the time-window constraints. As in the undirected case, the set of constraints (3.49) is added to obtain a strengthened formulation; they reduce the set of equivalent feasible solutions. Finally, the decision variables are defined in (3.50) and (3.51).

3.6.2 Model on the required arcs

This formulation is equivalent to the previous one; it reduces the size of the problem by considering only the network information related to the required arcs. Let the original problem be defined on the directed graph $G_d = (V_d, D)$ described in Section 3.6.1. If $\hat{R} \subset D$ is the set of

required arcs without the artificial arcs "a_l" and "a_e", the formulation on the required arcs is defined on the graph $\dot{G}_d = (\dot{N}_d, A_d)$ with $|\hat{R}|$ nodes in $\dot{N}_d$. Each node in $\dot{N}_d$ corresponds to a required arc in $\hat{R}$, and each arc in A_d represents the length of the shortest path in G_d between a pair of required arcs. $\dot{G}_d$ results in a complete graph.

Figure 3.4 shows an example of the transformation. In the original graph presented in a) the depot is located at the black node. The numbers in parentheses represent traversal costs, and the other numbers are the arc indices. There are three required arcs (2, 4, and 5). The graph in b) with three nodes is complete. The nodes are labeled with the same number as their respective required arcs, and the distances are indicated in parentheses.



a) Original graph b) Transformed graph

Figure 3.4: Transformed graph for the model on the required arcs

An artificial node "0" is added to represent the depot. We connect this node with each node in $\dot{G}_d$ by means of two arcs, one entering and one leaving the depot. Finally, each node in $\dot{G}_d$ adopts the time window corresponding to that of the arc that it represents.

In the transformed graph we look for a minimum-cost tour that starts and ends at the depot and satisfies the time-window constraints for each node. Given the parameters c_{ij}, T_{ij}, $[a_i\ b_i]$, and M_{ij}, and the decision variables x_{ij} and t_i defined in Section 2.2, the formulation is as follows:

$$\text{minimize} \sum_{i \in \dot{N}_d \cup \{0\}} \sum_{j \in \dot{N}_d \cup \{0\} | j \neq i} c_{ij} x_{ij} \tag{3.52}$$

s.t.:

$$\sum_{\substack{j \in \dot{N}_d \cup \{0\}| \\ j \neq i}} x_{ij} = 1 \quad \forall\ i \in \dot{N}_d \cup \{0\} \tag{3.53}$$

$$\sum_{\substack{i \in \dot{N}_d \cup \{0\}| \\ i \neq j}} x_{ij} = 1 \quad \forall\ j\ \in \dot{N}_d \cup \{0\} \tag{3.54}$$

$$t_i + T_{ij} \leq t_j + M_{ij}\left(1 - x_{ij}\right)\ \forall\ i \in\ \dot{N}_d\ , j \in\ \dot{N}_d\ \cup \{0\}\ / i \neq j \tag{3.55}$$

$$a_i \leq t_i\ \forall\ i\ \in \dot{N}_d\ \cup \{0\} \tag{3.56}$$

$$t_i \leq b_i\ \forall\ i\ \in \dot{N}_d \tag{3.57}$$

$$x_{ij} \in\ \{0,1\}\ \forall\ i,j\ \in\ \dot{N}_d\ \cup \{0\} | i \neq j \tag{3.58}$$

$$t_i \in \mathbb{R}^+\ \forall\ i\ \in \dot{N}_d \cup \{0\} \tag{3.59}$$

The formulation corresponds to the ATSP-TW. The objective is to minimize the total traversal cost. Inequalities (3.53) and (3.54) are the assignment constraints. The time-window constraints are (3.55), (3.56), and (3.57). Finally, the decision variables are defined by (3.58) and (3.59).

To solve this model we refer to Ascheuer et al. (2001), who solved the ATSP-TW using a branch-and-cut method with satisfactory results.

3.6.3 Tests

To evaluate the performance of the formulation on the arcs, we tested the model on a set of 144 random generated planar graphs labeled “setD”.The set of instances has between 4 and 100 nodes, 4 and 180 arcs, and 2 and 90 required arcs. We generate three kind of width of times windows $\{10,30,50\}$, as we did before.

The three largest instances (with the highest number of required arcs) solved to optimality (in less than three hours) have the following characteristics: $\{80, 142, 71, 10\}$,$\{100, 180, 18,30\}$, and $\{42, 71, 36, 50\}$ for respectively the number of nodes, the number of arcs, the number of required arcs, and the width of the time window. Table 3.6 shows the summary of results for this set of instances. *%|R|* represents the percentage of required arcs with respect to the total of arcs. *a* is the average number of arcs and the other notation of the Table 3.6 is the same as the Table 3.4.

Table 3.6: Model on the arcs – set of instances “setD”

TW	*%|R|*	*n*	*a*	*|R|*	*N*	*Sol*	*T*
	10	41.57	70.64	7.57	14	9	75.85
10	30	35.29	59.64	18.35	17	8	67.28
	50	35.29	59.41	29.94	17	7	2.23
	10	41.57	70.64	7.57	14	11	32.88
30	30	35.29	59.64	18.35	17	8	20.20
	50	35.29	59.41	29.94	17	6	86.64
	10	41.57	70.64	7.57	14	9	30.65
50	30	35.29	59.64	18.35	17	9	16.95
	50	35.29	59.41	29.94	17	9	64.93

We could solve a similar number of problems in each subset of instances grouped by width of the time window (10, 30, and 50). Our results indicate that the width of the time windows does not have an impact on the difficulty of the problems. Also the computational time seems not to be dependent on the width of the time windows. Detailed results for each instance are available at http://ftpprof.uniandes.edu.co/~pylo/inst/RPPTW/anex1-results.htm.

As it was mentioned before, our main interest is in the undirected case; the transformed model on the required arcs (Section 3.6.2) corresponds to a TSP *with time windows* and solving this formulation is the most efficient way to deal with the problem because again we obtain a formulation with less variables and a better bound. Our aim in this case is to evaluate the size of the problem that can be solved with the model on the arcs.

3.7 **Conclusions**

We have introduced several formulations for the undirected and directed RPPTW, and we have tested them on instances adapted from the literature and on a real network.

The results show that the models on the arcs and edges are not practical because they are too large and use the “big M.” For the directed case the model on the arcs could solve instances with up to 36 required arcs and wide time windows, and 71 required arcs with tight time windows. Neither number of optimal solutions or average of computational time seems to depend on the width of the time windows.

We propose two transformations for the undirected case. Results show that the model on the nodes is superior to the model on the required edges, i.e., it allows to solve to optimality twelve more instances and the running times are smaller.

In the undirected version of the problem, we exploited the formulation called “Model on the nodes”. The resulting problem is an ATSP-TW and side constraints. This model is solved using a cutting plane algorithm. We were able to solve to optimality 222 of 225 generated instances in less than two hours. We solved to optimality 10 of the 12 largest instances (60 nodes, 90 edges, and 45 required edges) in less than 15 minutes on average. Also we solved 5 of 9 instances of a real network with up to 104 required edges in less than 3.5 minutes.

For the directed case the problem is transformed into an equivalent ATSP-TW. Existing methods for large instances, such as cutting plane algorithms, can be used.

Future research could develop a branch-and-cut algorithm to solve the problem. It could be interesting to study efficient strategies to decide on which variables to branch and include other types of cuts. We also believe that metaheuristics should be explored to get good solutions in a short time when the dynamic case of the black-ice detection problem is considered, i.e., when the time windows or the road segments to visit vary over time.

References

Ascheuer, N., Fischetti, M., & Grötschel, M. (1999). Solving the Asymmetric Traveling Salesman Problem with Time Windows by branch-and-cut. Belin, Germany: Konrad-Zuse-Zentrun für informationstechnik Berlin.

Ascheuer, N., Fischetti, M., & Grötschel, M. (2001). Solving the Asymmetric Travelling Salesman Problem with time windows by branch-and-cut. *Mathematical Programming, 90*(3), 475-506. doi: 10.1007/PL00011432

Ascheuer, N., Jünger, M., & Reinelt, G. (2000). A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints. *Computational Optimization and Applications, 17*(1), 61-84. doi: 10.1023/A:1008779125567

Balas, E., Fischetti, M., & Pulleyblank, W. (1995). The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming, 68*(1-3), 241-265. doi: 10.1007/BF01585767

Christofides, N., Campos, V., Corberán, Á., & Mota, E. (1981). An algorithm for the Rural Postman Problem. London: Imperial College.

Corberán, A., & Sanchis, J. M. (1994). A polyhedral approach to the rural postman problem. *European Journal of Operational Research, 79*(1), 95-114. doi: http://dx.doi.org/10.1016/0377-2217(94)90398-0

Corberán, Á., & Sanchis, J. M. (1991). A polyhedral approach to the Rural Postman Problem (D. d. E. e. I. operative, Trans.). Spain: Universidad de Valencia.

Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters, 10*(1), 27-36. doi: http://dx.doi.org/10.1016/0167-6377(91)90083-2

Dror, M., Leung, J. Y., & Mullaseril, P. (2000). Livestock Feed Distribution and Arc Traversal Problems. In M. Dror (Ed.), *Arc Routing* (pp. 443-464): Springer US.

Eiselt, H. A., Gendreau, M., & Laporte, G. (1995). Arc Routing Problems, Part II: The Rural Postman Problem. *Operations research, 43*(3), 399-414. doi: doi:10.1287/opre.43.3.399

Fischetti, M., & Toth, P. (1997). A Polyhedral Approach to the Asymmetric Traveling Salesman Problem. *Management Science, 43*(11), 1520-1536. doi: 10.2307/2634585

Gueguen, C. (1999). *Méthodes de résolution exacte pour les problèmes de tournées de véhicules.* École Central Paris.

Jünger, M., Reinelt, G., & Rinaldi, G. (1995). Chapter 4 The traveling salesman problem. In T. L. M. C. L. M. M.O. Ball & G. L. Nemhauser (Eds.), *Handbooks in operations research and management science* (Vol. Volume 7, pp. 225-330): Elsevier.

Kang, M.-J., & Han, C.-G. (1998). Comparison of Crossover Operators for Rural Postman Problem with Time Windows. In P. K. Chawdhry, R. Roy, & R. K. Pant (Eds.), *Soft Computing in Engineering Design and Manufacturing* (pp. 259-267): Springer London.

Letchford, A. N., & Eglese, R. W. (1998). The rural postman problem with deadline classes. *European Journal of Operational Research, 105*(3), 390-400. doi: http://dx.doi.org/10.1016/S0377-2217(97)00090-8

Mullaseril, P. A. (1997). *Capacitated rural postman problem with time windows and split delivery.* (PhD.), University of Arizona, Arizona.

Mullaseril, P. A., & Dror, M. (1996). A set covering approach for directed node and arc routing problems with split deliveries and time windows: MIS Department, University of Arizona.

Mullaseril, P. A., Dror, M., & Leung, J. (1997). Split-Delivery Routeing Heuristics in Livestock Feed Distribution. *The Journal of the Operational Research Society, 48*(2), 107-116. doi: 10.2307/3010350

Nobert, Y., & Picard, J. C. (1994). *A heuristic algorithm for the Rural Postman Problem with Time Windows.* Paper presented at the ORSA/TIMS, Detroit.

Padberg, M., & Rinaldi, G. (1990). An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming, 47*(1-3), 19-36. doi: 10.1007/BF01580850

Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev., 33*(1), 60-100. doi: 10.1137/1033004

Wøhlk, S. (2005). *Contributions to arc routing.* University of Southern Denmark.

This article was published in:

Monroy-Licht, M., Amaya, C. A., & Langevin, A. (2014). The Rural Postman Problem with time windows. *Networks, 64*(3), 169-180. doi: 10.1002/net.21569

Preliminary results were presented at:

Monroy-Licht, M., Amaya, C.A., & Langevin, A. (2012) Modeling the rural postman problem with time windows. *25th European Conference on Operational Research*. 8-11 July, Vilnius, Lithuania

CHAPTER 4 ARTICLE 2 : ALNS FOR THE RURAL POSTMAN PROBLEM WITH TIME WINDOWS

Marcela Monroy-Licht[1,2], Ciro Alberto Amaya[3,4], André Langevin[1,2]

[1] Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal, Canada
[2]Centre de recherche sur les réseaux d'entreprises, la logistique et le transport (CIRRELT), Montréal, Canada

[3]Departamento de Ingeniería Industrial, Universidad de Los Andes, Bogotá, Colombia

[4]Grupo de investigación en producción y logística (PYLO), Bogotá, Colombia

Abstract

The *rural postman problem with time windows* (RPPTW) is the problem of serving some required edges with one vehicle; the vehicle must visit these edges during established time windows. This paper presents a competitive *adaptive large neighborhood search algorithm* (ALNS) to solve the problem. Computational experiments are performed on a large set of instances with up to 104 required edges. The results show that this approach is efficient, significantly reducing the computational time on large instances and achieving good solutions: the algorithm is able to solve to optimality 224 of 232 instances.

Keywords: Rural postman problem, Time windows, Adaptive large neighborhood search, Metaheuristics.

4.1 Introduction

Checking the status on main roads during winter is a daily responsibility that helps to prevent accidents. In the "Estrie," an administrative region of Quebec, this monitoring is conducted by the Ministry of Transportation from mid-October to mid-December. It is necessary to detect black ice in a timely fashion to carry out preventative measures. Black ice may appear at low temperatures close to 0°C or if the air warms suddenly after a prolonged cold spell. In the Estrie region, several weather stations report the meteorological conditions. Each morning, the patrol in charge of the monitoring plans the route to follow that day. The patrol uses the weather reports

from each zone and decides which roads are critical, i.e., have a high likelihood of black ice appearing.

To decide which route to follow, the patrol must solve a *rural postman problem* (RPP) with time constraints. Given a road network, it must visit some of the roads within a suitable time. This problem is known as the RPPTW, where the time windows are defined by the meteorological conditions reported.

More formally, given a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, if $E_R \subseteq E$ is a subset of required edges, the RPP involves finding a minimum-cost tour that visits all the edges in E_R at least once, starting and finishing at the same origin vertex. If the visits must be carried out during a defined time interval, the problem becomes the RPPTW; here, earliest and latest times are specified for the completion of the service of the required edges.

Other applications of the RPPTW may be found in areas related to arc routing problems such as postal deliveries and the treatment of roads with salt and grit to prevent them from freezing.

The RPPTW is NP-hard, even when $E_R = E$, since it reduces to the *Chinese postman problem with time windows*, which is NP-hard (Dror, 2000). If $E_R \subset E$ and all the time windows are defined by the interval $[\,0, \infty]$, it reduces to the RPP. The RPP has been shown to be NP-hard (Lenstra and Kan, 1976) .

The aim of this work is to present an algorithm that finds good, but not necessarily optimal, solutions. The method must solve large instances quickly, so we have developed a metaheuristic approach.

The remainder of this article is organized as follows. Section 4.1 summarizes related work. Section 4.2 gives a formal description of the problem and the solution method. Experimental results are presented in Section 4.3, and Section 4.4 provides concluding remarks.

4.2 Literature review

Arc routing problems with time windows have received less attention than the equivalent node routing problems. Corberán and Prins (2010) present the most recent survey of arc routing problems and their variants. The *capacitated arc routing problem with time windows* is the focus of several papers (Gueguen, 1999; Mullaseril, 1997; Reghioui et al., 2007; Wøhlk, 2005)

Another time-sensitive capacitated arc routing problem is the *arc routing problem with time dependent service costs* (Tagmouti et al., 2007).

Few researchers have explored the time-sensitive RPP. Letchford and Eglese (1998) present the RPP *with deadline classes*. They look for a minimum-cost route traversing a subset E_R of the edges of a graph, where E_R is divided into a number of deadline classes $R^1, R^2, \dots, R^L$. The edges in R^1 must be serviced by time T^1, those in R^2 must be serviced by time T^2, and so on. The authors propose a formulation, valid inequalities, and a cutting plane algorithm to solve problems with up to 50 vertices, 110 edges, and 7 R-connected components. They found optimal solutions for instances with up to 67 required edges.

The RPP *with time-dependent travel time* is presented by Tan and Sun (2011). The problem is defined on a directed time-dependent network $D = (V, A)$, where V is the vertex set, A is the arc set, and $A_R \subseteq A$ is the set of required arcs. The travel time D_{ij} for each arc $a_{ij} \in A$ is a function depending on the starting time t_i, i.e., $D_{ij}(t_i)$. The aim is to find a minimum-travel-time tour starting at the origin vertex v_0 and at starting time t_0, and traversing each required arc $a_{ij} \in A$ at least once. An integer linear programming formulation is proposed, and valid inequalities are given.

The first approach to the RPPTW is that of Nobert and Picard (1994). The authors define two types of required edges, E_1 and E_2. The edges in E_1 must be serviced during the morning and the edges in E_2 may be serviced at any time. A heuristic algorithm based on the solution of two rural path problems and on the computation of suitable penalties is presented, but no numerical results are given. Kang and Han (1998) solve a relaxed version of the problem: late arrivals are penalized. The authors present a genetic algorithm to solve a bi-objective problem that minimizes the total travel cost and the total penalty, and they compare three crossover operators.

Recently, Monroy-Licht et al. (2014) have presented several mathematical formulations for the RPPTW for the directed and undirected cases, as well as valid inequalities. The authors developed a cutting plane algorithm, which they tested on two sets of instances. For the first set, they solved 222 of 225 instances to optimality. The instances in the second set were larger, and they were able to solve 5 of 9 instances with up to 104 required edges.

The computational time becomes excessive when exact algorithms are used on large instances. We opted for an ALNS because it has performed well on time-window routing problems (Ropke and Pisinger, 2006a) and has achieved good results on complex arc routing problems (Riquelme-Rodríguez et al., 2014; Salazar-Aguilar, et al., 2012, 2013). The proposed ALNS addresses the undirected version of the RPPTW, but it could easily be extended to other versions of the problem.

4.3 Adaptive large neighborhood search

This section defines the RPPTW mathematically and describes the ALNS, which is implemented within a simulating annealing metaheuristic.

Let $G = (V, E)$, where V is the set of vertices, E the set of edges, and $E_R \subseteq E$ the set of required edges called *requirements*. Each edge i has an associated traversal cost c_i and a traversal time T_i. A time window $[a_i, b_i]$ is assigned to each required edge $i \in E_R$, where a_i is the earliest time and b_i the latest time to start the service at edge i. The RPPTW is the problem of finding a minimum-cost tour starting and finishing at the depot node and servicing each required edge on time. Several formulations of the problem are presented in Monroy-Licht et al. (2014).

ALNS was introduced by Pisinger and Ropke (2007) and Ropke and Pisinger (2006b). Basically, the algorithm constructs an initial solution and then attempts to improve it using competing removal and insertion heuristics. ALNS is based on two principles. First, it explores a large neighborhood of the solutions (since the algorithm has many possibilities for moving to other solutions, the neighborhood becomes very large). Second, it uses the best removal and insertion heuristics of the past iterations (at every iteration of the algorithm, the heuristics are selected based on their past performance).

4.3.1 Initial solution

A required edge is represented by the pair of nodes (i, j) corresponding to its end nodes. This notation indicates the direction in which the edge is traversed. We represent a solution x as the ordered sequence $(i,i)_0, (u,v)_1, \dots, (w,t)_n, (i,i)_{n+1}$ of the n required edges together with the edges placed at positions 0 and $n+1$, which represent the depot edge. The cost of the solution x

is the sum of the traversal cost across the sequence, i.e., the sum of the shortest paths between the final node of one required edge and the initial node of the next.

We run three simple heuristics based on sorting schemes to get an initial solution:

- Increasing release date (IRD): this heuristic sorts the required edges according to increasing release dates (lower bounds of time windows) and checks whether the sequence is feasible. Algorithm 1 gives the constructive procedure.
- Increasing due dates (IDD): This heuristic sorts the required edges according to increasing due dates (upper bounds of time windows) and checks whether this sequence is feasible. The constructive procedure is similar to Algorithm 1.
- Increasing center dates (ICD): This heuristic sorts the required edges according to increasing centers of the time windows, $a_i + \frac{b_i - a_i}{2}$, and checks whether this sequence is feasible. The constructive procedure is similar to Algorithm 1.

Algorithm 1: IRD

Input: Set of n required edges E_R
Output: x, an ordered sequence of required edges

1 $k = 0$
2 $x_0 \leftarrow (i, i)_k$
3 **While** $|E_R| \neq 0$ & x_0 is feasible **do**
4 $k = k + 1$
5 Choose $(u, v) \in E_R$ with the minimum release date
6 **If** $cost\big((w, t)_{k-1,}(u, v)\big) \leq cost\big((w, t)_{k-1,}(v, u)\big)$ **then**
7 $x_0 \leftarrow x_0 \cup (u, v)_k$, $E_R = E_R \setminus (u, v)$
8 **Else**
9 $x_0 \leftarrow x_0 \cup (v, u)_k$, $E_R = E_R \setminus (v, u)$
10 **End if**
11 **End while**
12 **If** $|x_0| = n + 1$ **then**
13 $x \leftarrow x_0$
14 **Else**
15 $x \leftarrow \emptyset$
16 **End if**

The best solution delivered by the three construction heuristics is chosen. If no feasible solution is found by any of the heuristics, we allow infeasible solutions by modifying line 3 of Algorithm 1 to *While* $|E_R| \neq 0 \ do$. We keep the resulting solution as the initial solution for the improvement phase.

4.3.2 Improvement phase

Given an initial solution x, we apply the ALNS algorithm until a stopping criterion is reached. The algorithm outputs the best solution x^* encountered during the search.

At each iteration, the algorithm chooses a removal and an insertion heuristic based on their weights. The selected removal heuristic deletes some required edges from the solution x and then, the insertion heuristic attempts to reinsert them in better positions. Seven removal and two insertion heuristics were implemented.

The weights of the heuristics are based on their historic performance. At each iteration we assign scores that depend on the performance (see Ropke and Pisinger (2006a)); a high score indicates a successful heuristic. If the removal and insertion heuristics are able to find a new overall best solution their scores are increased by σ_1. If the heuristics improve the current solution but not the overall best solution their scores are increased by σ_2. If the solution is worse than the previous solution but accepted with a given probability, the heuristic scores are increased by σ_3. The scores for both the removal and insertion heuristics are updated by the same amount, σ_1, σ_2, or σ_3.

The improvement phase is divided into 10 segments and 10 subsegments. At the beginning of each segment the weights are set to one. After each subsegment of 250 iterations, the weights are updated according to the scores obtained. Let $\overline{w}_{h,l}$ be the weight of heuristic h used in subsegment l. The weight for the heuristic h to be used in subsegment $l+1$ is calculated as follows:

$$\overline{w}_{h,l+1} = r\frac{\pi_h}{\theta_h} + (1-r)\frac{\sum_{d=1}^{l-1}\overline{w}_{h,d}}{l-1} \tag{4.1}$$

π_h is the score of heuristic h obtained during the previous subsegment l; θ_h is the number of times that heuristic h was used in subsegment l; r is a factor, $0 \leq r \leq 1$, that controls the emphasis placed on the score obtained in the last subsegment with reference to the historical average weight.

Each removal/insertion heuristic h is chosen at each iteration by two separate roulette-wheel mechanisms. The probability of selecting heuristic *h* for subsegment l is

$$\frac{\bar{w}_{h,l}}{\sum_{h=1}^{o} \bar{w}_{h,l}} \tag{4.2}$$

where o is the total number of removal/insertion heuristics (i.e., o=7 or 2).

Let $C(x)$ represent the cost of the current solution x. The new solution given by the ALNS at each iteration is represented by $\bar{x}$. A simulated annealing mechanism determines whether or not the new solution $\bar{x}$ should be accepted. A solution is accepted if $C(\bar{x})$ is less than $C(x)$, and it is accepted with probability $e^{\frac{-C(\bar{x})-C(x)}{T}}$ if $C(\bar{x})$ is greater than or equal to $C(x)$, where $T > 0$ is the temperature. The temperature has an initial value T_0 calculated from the initial solution so that a solution that is ω% worse than the current solution is accepted with 50% probability (Ropke & Pisinger, 2006b). At each subsegment the temperature is decreased slightly: $T_{l+1} = T_l - (\frac{T_0}{100})$, where T_{l+1} is the temperature of subsegment $l + 1$.

Toward the end of the search we accept only good moves, and therefore it is harder for the heuristics to get high scores. If no solutions improve the current best solution in an entire subsegment, the algorithm perturbs the best known solution by reversing the traversal direction of $f|E_R|$ required edges. The $f|E_R|$ required edges are randomly chosen.

In our implementation we stopped the algorithm after 25,000 iterations. Our ALNS also allows infeasible solutions (with regards to the time windows), which improves the overall search; the infeasibility is penalized with a high cost in the objective function.

4.3.2.1 Removal heuristics

Our ALNS framework for the RPPTW uses seven different removal heuristics. These heuristics take as input a given solution x and output a random number $q = \lceil [p_1|E_R|, p_2|E_R|] \rceil$ of required edges, called *requests*, which are removed from the tour.

R1. Random removal: This simple heuristic selects q requests at random and removes them from the solution x. The purpose of this heuristic is to diversify the search.

R2. Series removal: This heuristic selects a series of q consecutive requests. It randomly selects a position k between $k = 1$ and $k = n - q + 1$, where n represents the number of required edges, and removes the next q requests starting at position k and finishing at position $k + q - 1$.

R3. Worst-removal: This heuristic chooses q requests that are very expensive, i.e., with a long distance to cover them. It seems reasonable to try to remove requests with a high cost and insert them elsewhere to obtain a better solution.

Given a request (i, j) placed at position k in solution s, we define the cost of the request as $cost((i,j)) = d((u,v)_{k-1}, (i,j)_k) + d((i,j)_k, (w,t)_{k+1})$, where d represents the distance function between two requests.

The worst-removal heuristic repeatedly chooses the request (i, j) with the largest $cost((i,j))$ until h $(h = 2q)$ requests have been selected. To obtain variability in the removal, the heuristic is randomized: q random requests are removed from the h selected requests.

R4. Space-related removal: This heuristic removes a set of q related requests that could be closer and hence provide a better solution. For the RPPTW, we define the relatedness $r_{\{i,j\}\{u,v\}}$ of two requests by the distance (the shortest path) between them.

The heuristic initially selects a request (i, j) at random and places it into the set Q of selected requests. Then it repeatedly calculates $r_{(i,j)(u,v)}$ for all (i, j) in Q and (u, v) not in Q, and chooses the request (u, v) that minimizes $r_{(i,j)(u,v)}$. The algorithm stops when q requests have been chosen. Ties are broken randomly to diversify the output.

R5. Time-related removal: This heuristic is a variant of the space-related removal heuristic. Here we try to remove requests that are close in space and have similar time windows. The motivation is to try to remove requests that are easy to interchange.

We measure the time-window relatedness $m_{(i,j)(u,v)}$ as the difference of the centers of the time windows. The center of request (i, j) is defined by $m_{(i,j)} = a_i + \frac{b_{(i,j)} - a_{(i,j)}}{2}$. The heuristic first selects h requests $(h = 2q)$ as in the space-related removal heuristic. Then, it chooses the q requests in Q with the minimum $m_{(i,j)(u,v)} = a_{i,j} + \frac{b_{(i,j)} - a_{(i,j)}}{2}$ - $a_{u,v} + \frac{b_{(u,v)} - a_{(u,v)}}{2}$, i.e., it iteratively chooses the pair $(i, j), (u, v)$ that minimizes $m_{(i,j)(u,v)}$ and places this pair into the set of selected requests.

R6. Far-time-close-position removal: This heuristic removes requests that seem to be placed wrongly because they are close in the solution but they have a high time windows center difference.

The heuristic starts by selecting a request (i, j) at random. Then the algorithm identifies the h requests with the largest differences in their time-window centers with respect to the time-window center of request (i, j). These requests are placed into H. Then, the heuristic iteratively chooses the $q - 1$ requests in H closest to (i, j) in the solution x. The set of requests to remove is completed by adding the request (i, j).

R7. Series-far-time removal: This heuristic is an extension of the series removal heuristic. The purpose of this heuristic is to choose requests that could be placed wrongly, i.e., they are close in the solution but they have a large mean time-window difference.

The heuristic first selects a series of $h = 2q$ requests, as in the series removal heuristic, and places them into H. Then, it chooses the q requests with highest $m_{(i,j)(u,v)}$ for $(i, j), (u, v)$ in H.

4.3.2.2 Insertion heuristics

The insertion heuristics construct a solution $\bar{x}$ by inserting requests into the partial solution x' obtained when the q requests from the removal heuristics are removed from x. The removed requests are initially placed in a set U.

I1. Basic greedy insertion: This heuristic is a simple construction heuristic that repeatedly inserts a request (i, j) into the cheapest position considering both directions in which the request can be inserted.

We define $c_{(i,j),k} = d\big((u, v)_{k-1}, (i, j)\big) + d\big((i, j), (w, t)_k\big) - d((u, v)_{k-1}, (w, t)_k)$ as the insertion cost of request (i, j) at position k of solution x'. If (i, j) cannot be inserted at position k, then $c_{(i,j),k} = \infty$. Let $\Delta x'_{(i,j)}$ denote the change in the objective value of solution x' incurred by inserting request (i, j) into its best overall position, i.e., $\Delta x'_{(i,j)} = min_k\, c_{(i,j),k}$.

At each iteration the algorithm chooses the request $(i, j) \in U$ with the minimum $\Delta x'_{(i,j)}$ and inserts (i, j) into its cheapest position. Then (i, j) is removed from U. This process

continues until all the requests in U have been inserted into solution x', and we then set $\bar{x} = x'$.

This heuristic has an obvious drawback: it often postpones the placement of "hard" requests (requests with large $\Delta x'_{(i,j)}$) until the last iterations, when there are few feasible positions because of the time windows. The next heuristic tries to circumvent this problem.

I2. Regret insertion: The regret heuristic tries to improve on the basic greedy insertion heuristic by incorporating a kind of look-ahead information when selecting the request to insert. We define the regret value $r_{(i,j)}$ as the difference of the two lowest values $c_{(i,j),k}$ for each request $(i,j) \in U$. In other words, the regret value is the difference between the cost of inserting the request into its best position and that of its second-best position.

At each iteration the regret heuristic chooses to insert the request $(i,j) \in U$ with the maximum $r_{(i,j)}$. The selected request is inserted into its cheapest position considering both directions in which the request can be inserted. The process stops when all requests in U have been inserted into solution x', and we then set $\bar{x} = x'$.

4.4 Results

In this section we describe our computational experiments. We test the performance of the algorithm against the cutting plane algorithm proposed by Monroy-Licht et al. (2014). There are four major objectives for this section:

1. We present the set of instances and the optimal value, not previously reported, for some instances.
2. We discuss the parameter tuning based on statistical tests.
3. The performance and robustness of the overall algorithm depends on the choice of the removal and insertion procedures (Ropke and Pisinger, 2006a). Therefore, the third objective is to compare the performance of the heuristics.
4. We also explore the performance of the algorithm on larger instances.

The tests were coded in Python and run on a 1.9-GHz AMD PC with 1983 MB of internal memory. We chose this test environment since it is the same as that used in (Monroy-Licht et al., 2014).

4.4.1 Instances

We tested the algorithm on two sets of instances, both proposed by Monroy-Licht et al. (2014). These are the only known benchmark instances for the RPPTW.

The first group (*set1*) is a set of 225 instances with between 2 and 45 required edges and 3 types of time windows: tight, labeled TW=10; intermediate, TW=30; and wide, TW=50. The second set of instances (*set2*) is generated for a real network of major roads in the Estrie region. This set of 8 instances has between 74 and 104 required edges and 3 widths of time windows (10, 30, and 50). Monroy-Licht et al. (2014) presented 9 instances in *set2*, but we did not use instance *Inst-00* because it seems to have an error in the data.

In *set1*, each required edge has a randomly generated time window. The instances in *set2* have a special feature: the time windows are generated as four or five time slots, and so several required edges have the same time windows.

Monroy-Licht et al. (2014) solved 227 instances to optimality. We ran the cutting plane algorithm proposed by the authors for the instances that could not be solved in less than 3 hours in (Monroy-Licht et al., 2014). With a time limit of 10 hours and a parameter tolerance gap of 0 we solved to optimality 5 new instances. Table 4.1 presents the optimal solution values (*O.F.*) and the time in seconds taken by the cutting plane algorithm. Column *R* indicates the number of required edges, and column *TW* indicates the type of time windows.

The cutting plane algorithm spends significantly more time on TW-B60A115 and TW-C40C152 than on the other instances. Since Inst-02, Inst-04, and Inst-05, which belong to *set2*, are larger than TW-B60A115 and TW-C40C152, we can see that it is easier to solve the larger instances because the structure of their time windows is different, as explained above.

Table 4.1: Optimal solutions for benchmark instances

Instance	*R*	*TW*	*O.F.*	*Time*
TW-B60A115	45	30	511.0	25779.5
TW-C40C152	35	50	289.0	23729.6
Inst-02	74	50	259.7	366.2
Inst-04	104	30	289.2	615.4
Inst-05	104	50	289.2	306.9

4.4.2 Tuning set parameters

This section determines the parameters that need to be tuned. The weights of the ALNS are controlled by parameters $\sigma_1, \sigma_2, \sigma_3$ (performance scores of removal and insertion heuristics) and r (emphasis on the score obtained in recent iterations). To control the acceptance criteria we use the parameter ω. Finally, we must determine p_1 and p_2, the parameters that control how many requests we remove and insert, and f, which controls the number of required edges that can be reversed (change of the traversal direction).

Intuitively, p_2 and f should have a more significant effect on the quality of our algorithm. We focused on the calibration of these two parameters. For the other parameters we decided to use the values used in similar work in the literature. For all the experiments we used the parameter values determined in (Pisinger and Ropke, 2007) and (Ropke and Pisinger, 2006b) for parameters r and ω. They were respectively set to 0.7 and 5. The parameters σ_1, σ_2, and σ_3 were set to 15, 8, and 2 respectively. The relative values of these parameters maintain the relationship determined in (Pisinger and Ropke, 2007; Ropke and Pisinger, 2006b). p_1 was set to 0.1 as in (Pisinger and Ropke, 2007).

The maximum number of requests that can be removed in a single iteration, $p_2|E_R|$, and the number of required edges that can be reversed (change of the traversal direction), $f|E_R|$, are controlled by p_2 and f, which are expressed as a percentage of the number of required edges.

We produced a fair parameter setting through empirical tests. Two values were chosen for p_2: 0.2 and 0.4. Parameter f was set to 0.02, 0.05, and 0.1. Then we applied a statistical *F*-test with

two factors (p_2 and *f)*; p_2 had 2 levels and *f* had 3 levels. The statistical test of factorial crossing gives us information about the effect of the six statistical treatments. Each statistical treatment was run 5 times.

For the tuning tests, we selected a random sample of 30 instances with different sizes and time windows widths from *set1* and *set2*; all of them with known optimal solutions. At the 95% level the test rejected the hypothesis that there is a difference between the performance of the parameter combinations, with p-values of 0.7093 and 0.6005 for the effect of *f* and p_2, and 0.677 for the effect of the interaction between *f* and p_2. Therefore, the ALNS results are sufficiently robust with respect to variations of the parameters, and no significant improvement in the solution can be expected by selecting a different parameter combination p_2, *f*. For the subsequent tests, the parameters p_2 and f were set to 0.2 and 0.1 respectively, since the preliminary tests indicated that these values provided the best performance.

4.4.3 Performance of removal and insertion heuristics

The objective of this section is to compare the individual performance of each removal and insertion heuristic to that of all the heuristics. We then explore the best combination of heuristics.

As before, we denote the removal heuristics by R1, R2, R3, R4, R5, R6, and R7 and the insertion heuristics by I1 and I2. We implemented different versions of the ALNS. Each was applied five times to each instance (and all subsequent experiments were also performed five times). Initially we applied VR1, VR2, VR3, VR4, VR5, VR6, VR7, VI1, and VI2. Versions VR1 to VR7 involve using one removal heuristic and both insertion heuristics I1 and I2. Versions VI1 and VI2 involve using all seven removal heuristics and one insertion heuristic. We refer to these as *single versions*. We also applied the full version (FV) of the ALNS, i.e., with all removal and insertion heuristics.

Table 4.2 summarizes the performance of the different versions on *set1*. Detailed results for each instance and the full experimental results can be found at http://wwwprof.uniandes.edu.co/~pylo/inst/ALNSRPPTW/instances.html.

Table 4.2 shows that the most efficient removal versions are VR2, VR1, and VR4: they found the optimal solution for at least 92% of times. Version VR5 has the worst performance, finding the optimal solution for just 57% of times. All the single removal versions except VR5

outperform FV. Version VR7 performs well for instances with fewer than 35 required edges. The columns showing the single insertion versions indicate that version VI2 outperforms VI1 in most of the cases: VI2 was effective for 82% of times and version VI1 for 62%. The performance of FV was similar to that of VI2.

Table 4.2: Performance of single versions of the ALNS on *set1*

R	**TW**	***n.e.***	**VR1**	**VR2**	**VR3**	**VR4**	**VR5**	**VR6**	**VR7**	**VI1**	**VI2**	**FV**
	10	215	**215**	**215**	**215**	**215**	181	**215**	**215**	**205**	204	204
≤10	30	215	212	212	202	211	154	**214**	207	164	**188**	190
	50	215	**205**	**205**	204	**205**	163	200	203	160	**195**	194
	10	45	**45**	**45**	**45**	**45**	32	**45**	**45**	44	**45**	**45**
12	30	45	**45**	**45**	**45**	**45**	17	44	**45**	27	**41**	37
	50	45	42	44	37	**45**	11	34	35	14	**33**	33
	10	15	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
16	30	15	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
	50	15	14	**15**	10	14	5	**15**	**15**	5	**9**	11
	10	30	**30**	**30**	29	**30**	14	25	**30**	24	**26**	29
21	30	30	**30**	**30**	**30**	29	7	29	29	5	**17**	19
	50	30	**30**	28	27	28	1	23	**30**	0	**20**	20
	10	20	**15**	**15**	**15**	**15**	5	**15**	**15**	3	**14**	13
27	30	20	**20**	**20**	15	**20**	6	15	**20**	0	**16**	13
	50	20	10	**19**	6	15	0	2	7	0	**10**	11
	10	30	**28**	27	23	25	5	18	25	8	**27**	24
35	30	30	16	**17**	10	10	1	8	11	0	**10**	10
	50	30	24	**30**	15	26	5	16	22	5	**20**	18
	10	20	**14**	6	10	9	1	7	8	0	**7**	6
45	30	20	9	**11**	2	9	0	1	7	0	**2**	3
	50	15	4	**8**	2	6	0	0	1	0	**5**	2
	Total	1120	1038	1052	972	1032	638	956	1000	694	919	912

The table compares the different single versions of ALNS and the full version. The first column shows the number of required edges; the second gives the width of the time windows. Column *n.e.* indicates the number of experiments performed for a data set of size *R* (number of required edges). The other columns indicate how many times the optimal solution was reached by the corresponding version of the ALNS. Bold entries indicate that the version solved a maximum number of problems to optimality (for single removal and single insertion versions respectively). The last row shows the number of times the optimal solution was found over all the experiments by each version.

The gaps are shown in Table 4.3. The gap is averaged over all the experiments of a given time-window width. We also calculate the average gap for instances that did not reach the optimal value. The last three rows of the table indicate the maximum gap over all the experiments for each single version of the ALNS.

Table 4.3: Gaps for single versions of the ALNS on *set1*

Gap	***TW***	**VR1**	**VR2**	**VR3**	**VR4**	**VR5**	**VR6**	**VR7**	**VI1**	**VI2**	**FV**
Average gap (%)	10	**0.080**	0.154	**0.130**	0.145	1.154	0.165	**0.123**	0.635	**0.232**	0.249
	30	**0.124**	**0.075**	0.568	**0.172**	2.851	0.461	0.242	2.754	**0.931**	0.924
	50	**0.336**	**0.206**	0.494	**0.231**	4.201	1.062	0.675	4.763	**0.774**	0.922
Gap (%) for instances that did not reach the optimal value	10	2.319	2.633	**2.126**	2.584	3.546	**1.770**	**2.099**	3.133	**2.348**	2.398
	30	**1.659**	**1.121**	3.806	**1.793**	6.109	3.526	2.218	6.298	**4.059**	3.936
	50	**3.035**	3.637	**2.649**	**2.755**	8.403	4.912	4.379	9.476	**3.673**	4.212
Maximum gap (%)	10	**3.753**	5.622	**3.753**	4.836	10.526	**3.753**	4.771	7.631	**7.609**	7.609
	30	**3.644**	**2.963**	13.889	**4.815**	21.875	8.713	6.061	19.149	**18.750**	14.063
	50	**8.333**	**8.858**	9.091	**8.333**	34.146	15.287	15.873	31.746	**18.182**	26.829

The bold entries indicate the three best gaps for the single removal versions and FV. The bold entries in columns VI1 and VI2 indicate the better gap for the two single insertion versions.

The average gap was less than 1.1% for all the single removal versions except VR5. VR1, VR2, and VR4 had the smallest average gaps; these gaps were always better than those of FV. Versions VR1, VR2, and VR4 had the smallest maximum gaps on the hardest instances (TW=50). For the insertion versions, VI2 achieved better gaps than VI1 in all cases.

Tables 4.4 and 4.5 present the results of other tests. We applied the ALNS with the three best removal heuristics: R1, R2, and R4. Version VR124I2 includes I2 as the insertion heuristic, and VR124I12 includes both I1 and I2. Table 4 lists how many times these versions found the optimal solution for all the instances.

Version VR124I12 performs better than VR12I2. Note that although VR2 outperforms VR1, the algorithm obtains better results when both insertion heuristics are included. Overall, VR124I12 is the best version. It finds the optimal solution for 95.9% (1074/1120) of times.

Table 4.5 lists the gaps for VR124I2 and VR124I12. It shows that VR124I12 is quite stable: the average gap never exceeds 0.2% and it has the smallest average gap of all versions of the ALNS.

Table 4.4: Performance of versions VR124I2 and VR124I12 on *set1*

R	***TW***	***n.e.***	**VR124I2**	**VR124I1I2**
	10	215	**215**	**215**
≤10	30	215	210	**214**
	50	215	**205**	**205**
	10	45	**45**	**45**
12	30	45	**45**	**45**
	50	45	42	44
	10	15	**15**	**15**
16	30	15	**15**	**15**
	50	15	**15**	**15**
	10	30	**30**	**30**
21	30	30	**30**	**30**
	50	30	29	**30**
	10	20	**15**	**15**
27	30	20	**20**	**20**
	50	20	12	18
	10	30	29	**30**
35	30	30	13	**21**
	50	30	23	29
	10	20	13	13
45	30	20	14	**15**
	50	15	7	**10**
	Total	1120	1042	1074

The table should be interpreted like Table 2. Bold entries indicate when the version solved a maximum number of problems to optimality over all versions of the ALNS (VR1 to VR7, FV, VI1, VI2, VR124I2, and VR124I12).

Table 4.5: Gaps for VR124I2 and VR124I12 on *set1*

Gap (%)	***TW***	**VR124I2**	**VR124I1I2**
Average	10	0.098	**0.078**
	30	0.302	**0.043**
	50	0.387	**0.163**
Instances that did not reach the optimal value	10	2.816	2.447
	30	4.044	**1.071**
	50	3.867	3.183
Maximum	10	**3.753**	**3.753**
	30	13.889	**1.887**
	50	15.873	**8.333**

The interpretation is as for Table 4.3. Bold entries indicate when the corresponding version obtained the smallest gap over all versions (VR1 to VR7, FV, VI1, VI2, VR124I2, and VR124I12).

Finally, we compared the performance of the best version with that of the cutting plane algorithm (Monroy-Licht et al., 2014). The results are summarized in Table 4.6. VR124I12 fails to find the optimal solution for only 6 instances. Moreover, the computational time of the cutting plane algorithm increases significally on the hardest instances: ALNS solves the hardest problems in less than 5.5 min. on average while the cutting plane algorithm requires 34.3 min.

Table 4.6: Comparison of VR124I12 and cutting plane algorithm

TW		**10**					**30**					**50**				
		VR124I12			***Cutting plane***		**VR124I12**			***Cutting plane***		**VR124I12**			***Cutting plane***	
R	***ins***	***sol***	***Avg-t***	***Max-t***	***Avg-t***	***Max-t***	***sol***	***Avg-t***	***Max-t***	***Avg-t***	***Max-t***	***sol***	***Avg-t***	***Max-t***	***Avg-t***	***Max-t***
≤10	43	43	7.35	11.50	**0.13**	**0.83**	43	7.65	12.61	**0.18**	**0.79**	41	7.80	12.23	**0.27**	**0.98**
12	9	9	20.22	22.04	**0.51**	**0.81**	9	22.01	24.74	**1.60**	**2.85**	9	22.98	25.24	**1.27**	**1.99**
16	3	3	31.86	34.25	**1.68**	**2.31**	3	36.03	38.69	**3.32**	**7.27**	3	36.19	39.77	**4.70**	**6.53**
21	6	6	58.49	64.60	**4.26**	**9.65**	6	63.64	69.84	**6.74**	**9.14**	6	68.05	75.19	**10.34**	**33.70**
27	4	3	87.52	96.44	**8.44**	**10.75**	4	95.59	103.92	**16.36**	**34.08**	4	**99.20**	**109.10**	1484.43	5494.11
35	6	6	145.36	175.01	**29.14**	**46.00**	5	151.51	179.98	**44.44**	**83.02**	6	**174.50**	**187.26**	6110.53	23729.59
45	4	3	257.88	287.62	**44.63**	**60.23**	3	**301.05**	**332.02**	6892.40	25779.46	4	**306.25**	**335.30**	2060.24	6030.17

The table gives the number of problems solved to optimality and the computational times. Column *ins* shows the number of instances of size *R*. The data set is divided into three groups: TW=10, TW=30, and TW=50. For each group we report how many problems were solved to optimality (*sol*) and the average (*Avg-t*) and maximum (*Max-t*) ALNS computational time. The columns for the cutting plane algorithm show the average and maximum computational times reported by Monroy-Licht et al. (2014). Bold entries indicate the faster method.

4.4.4 Results for *set2*

Our final experiments were carried out on 8 larger instances based on a real network (Monroy-Licht et al., 2014). We applied only VR124I12, because it had good results in the previous tests.

Because we are interested in an algorithm that works well quickly, we analyze the quality of the solutions in terms of the running times. We also analyze the effect of the parameter *q*, which has the largest impact on the solution time: when more requests can be removed in a single iteration the solution time is higher.

We applied the ALNS with different values of *q*, and we used different limits on the computational time. We set q = 3, 5, 10, 40 or a random number between $0.1|R|$ and $0.2|R|$. We set the time limit to 180, 600, 900, and 1800 s. We applied the ALNS five times for each instance

and each combination of q and the time limit. We stopped the algorithm when it reached 25,000 iterations or the time limit.

Table 4.7 presents the average gap for all the experiments. The average gap is not reported if the algorithm uses less time to reach 25,000 iterations than the time limit; we reallocated those experiments to a group based on the upper bound of their running times. For q = 5 the algorithm obtained the best average gap in 4 instances, using a maximum of 685 s. For q = 10 and a time limit of 1800 s, we obtained the smallest gaps for 4 instances.

Table 4.7: Average gap: Study of the effect of q and *time limit*

		Time limit				***Avg. gap -***
Instance	***q***	**180**	**600**	**900**	**1800**	***all time limits (%)***
	random	3.363	2.814	2.688	2.281	2.793
Inst-01	3	4.585	4.368			4.422
$\|R\| = 74$	5	3.627	2.922			3.098
$TW = 30$	10	3.591	3.148	3.174	2.969	3.220
	40	34.355	18.820	9.997	7.975	17.787
Avg. gap - all q (%)		9.904	5.183	5.230	4.456	6.264
	random	4.317	3.641	3.594	3.272	3.706
Inst-02	3	4.347	4.412			4.396
$\|R\| = 74$	5	3.838	3.630			3.682
$TW = 50$	10	4.190	3.695	3.585	3.219	3.672
	40	10.980	7.671	7.275	5.556	7.871
Avg. gap - all q (%)		5.535	4.348	4.818	4.016	4.665
	random	9.232	7.143	5.121	5.212	6.677
Inst-06	3	5.702	3.766			4.250
$\|R\| = 93$	5	5.942	2.854			3.626
$TW = 10$	10	9.633	5.461	3.806	3.516	5.604
	40	21.062	15.696	13.904	11.851	15.505
Avg. gap - all q (%)		10.169	5.427	7.467	6.968	7.132
	random	4.341	3.361	2.646	1.613	2.990
Inst-07	3	3.235	3.213			3.219
$\|R\| = 93$	5	2.853	1.905			2.142
$TW = 30$	10	3.809	1.891	1.399	0.528	1.907
	40	12.253	7.577	7.205	5.176	7.897
Avg. gap - all q (%)		5.108	3.196	3.750	2.439	3.631

Table 4.7: Average gap: Study of the effect of q and *time limit*

		Time limit				***Avg. gap -***
Instance	***q***	**180**	**600**	**900**	**1800**	***all time limits (%)***
	random	4.582	2.509	2.485	1.732	2.827
Inst-08	3	4.544	3.645			3.870
$\|R\| = 93$	5	3.122	2.467			2.631
$TW = 50$	10	4.253	2.596	1.726	0.851	2.357
	40	9.232	7.315	4.834	4.701	6.486
Avg. gap - all q (%)		5.091	3.446	2.974	2.478	3.634
	random	9.682	5.519	6.837	5.252	6.823
Inst-03	3	4.159	2.965			3.264
$\|R\| = 104$	5	2.548	1.472	1.470		1.740
$TW = 10$	10	6.126	3.909	3.141	2.461	3.910
	40	20.702	15.902	16.507	14.524	16.909
Avg. gap - all q (%)		8.643	4.358	8.368	7.413	6.529
	random	3.568	2.743	2.414	2.155	2.720
Inst-04	3	2.583	2.247			2.331
$\|R\| = 104$	5	1.421	0.831	0.884		0.983
$TW = 30$	10	2.595	1.861	1.593	0.984	1.758
	40	17.825	13.789	11.801	10.634	13.378
Avg. gap - all q (%)		5.263	3.317	4.831	4.591	4.234
	random	5.534	4.770	3.758	2.519	4.145
Inst-05	3	2.742	2.691			2.704
$\|R\| = 104$	5	1.753	1.266	0.968		1.258
$TW = 50$	10	4.170	2.676	1.932	1.646	2.606
	40	20.196	14.820	12.748	11.986	14.938
Avg. gap - all q (%)		6.879	4.396	4.250	5.384	5.130
Avg. gap - all instances (%)		**7.085**	**4.205**	**5.167**	**4.719**	**5.153**

The columns are as follows. Instance gives the name and description of each instance, q is the number of requests to remove and insert in each iteration, and the next four columns are the maximum running time for each time limit. The final column is the average gap for all experiments with the corresponding value of q. The table also reports for each instance the average gap over all experiments with a given time limit. The last row of the table gives the average gap over all instances with a given time limit.

Table 4.8 summarizes the best results obtained by the algorithm in each instance. The first three columns present the features of each instance. The value of the best solution obtained by the algorithm is presented in column *ALNS*. The corresponding values of q and the time limit are given in the column *Best combination*. The average computational time in seconds is presented in

column *T*. The number of times the algorithm obtained the optimal solution is given in column *n*. The last columns give the optimal value, the computational time in seconds for the cutting plane algorithm (Monroy-Licht et al., 2014), and the gap. Note that in some cases the running time *T* is smaller than the time limit because the algorithm has reached the limit on the number of iterations. Although the running times of the exact method are in general better, the metaheuristic finds good solutions; it solves six instances to optimality, and the minimum gap for the other two is less than 0.6%.

Table 4.8. Best solutions: *set2*

Instance	***R***	***TW***	***ALNS***	***Best combination q, time limit***	***T***	***N***	***OF****	***t****	***Gap (%)***
Inst-01	74	30	259.7	random,600	600	1	259.7	**165.46**	0.00
				random, 1800	1654	1			
Inst-02	74	50	261.2	5,1800	456	0	259.7	**366.17**	0.59
Inst-06	93	10	301.0	5, 600	545	0	299.7	**193.41**	0.45
Inst-07	93	30	299.7	random, 1800	1800	1	299.7	**208.75**	0.00
				5, 600	499	1			
				10, 1800	1695	4			
Inst-08	93	50	299.7	random, 600	600	1	299.7	**137.76**	0.00
				random, 1800	1800	3			
				10, 1800	1678	1			
Inst-03	104	10	289.2	5, 180	**180**	1	289.2	202.29	0.00
				5, 900	582.50	2			
				5, 1800	555.66	3			
Inst-04	104	30	289.2	5, 600	585.8	5	289.2	615.35	0.00
				5, 900	614	1			
				5, 1800	**580.25**	4			
Inst-05	104	50	289.2	5, 600	600	2	289.2	**366.88**	0.00
				5, 900	641	3			
				5, 1800	632	2			

Bold entries indicate the faster method.

4.4.5 Summary of computational results

We have applied different versions of the ALNS to compare the performance of seven removal and two insertion heuristics and a full version that includes all of them. The selection of the removal heuristic has an impact on the solution quality: the removal heuristics based on randomness (R1 and R2) and the one that considers distance requirements (R4) perform better than the others. I2, the insertion heuristic that prioritizes the most difficult insertions, outperforms I1, which prioritizes the cheapest insertions.

The most efficient version is VR124I12; it uses R1, R2, and R4 as removal heuristics and I1 and I2 as insertion heuristics. It was able to find 218 optimal solutions for *set1* and 6 for *set2*.

The results show that the ALNS performs well; for the hardest instances (TW=50) the average gap of VR124I12 was 0.163%, and it required less than 336 s in the worst case; the exact method takes 20 times as long.

For larger instances, the number of requests that can be removed and inserted at each iteration has an important impact on the solution time. For $q = 5$ the average gap is 2.39% and the maximum running time is 685 s.

Although the instances in *set2* have more required edges than those in *set1,* the latter are harder to solve because of their time-window structure.

4.5 Conclusions

We have proposed an ALNS algorithm for the RPPTW. We have proposed and evaluated seven removal heuristics and two insertion heuristics. The best results were achieved when the algorithm used a combination of the best removal heuristics.

The ANLS is robust and performs well compared to exact methods. It found 224 optimal solutions for the 232 benchmark instances while significantly reducing the computational time on the hardest instances.

It would be interesting to apply the ideas presented here to other arc routing problems with time windows, taking advantage of the flexibility of the algorithm.

References

Corberán, A., & Prins, C. (2010). Recent results on Arc Routing Problems: An annotated bibliography. *Networks, 56*(1), 50-69. doi: 10.1002/net.20347

Dror, M. (2000). *Arc routing: theory, solutions and applications*: Kluwer Academic Publishers.

Gueguen, C. (1999). *Méthodes de résolution exacte pour les problèmes de tournées de véhicules.* École Central Paris.

Kang, M.-J., & Han, C.-G. (1998). Comparison of Crossover Operators for Rural Postman Problem with Time Windows. In P. K. Chawdhry, R. Roy, & R. K. Pant (Eds.), *Soft Computing in Engineering Design and Manufacturing* (pp. 259-267): Springer London.

Lenstra, J. K., & Kan, A. H. G. R. (1976). On general routing problems. *Networks, 6*(3), 273-280. doi: 10.1002/net.3230060305

Letchford, A. N., & Eglese, R. W. (1998). The rural postman problem with deadline classes. *European Journal of Operational Research, 105*(3), 390-400. doi: http://dx.doi.org/10.1016/S0377-2217(97)00090-8

Monroy-Licht, M., Amaya, C. A., & Langevin, A. (2014). The Rural Postman Problem with time windows. *Networks, 64*(3), 169-180. doi: 10.1002/net.21569

Mullaseril, P. A. (1997). *Capacitated rural postman problem with time windows and split delivery.* (PhD.), University of Arizona, Arizona.

Nobert, Y., & Picard, J. C. (1994). *A heuristic algorithm for the Rural Postman Problem with Time Windows.* Paper presented at the ORSA/TIMS, Detroit.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34*(8), 2403-2435. doi: http://dx.doi.org/10.1016/j.cor.2005.09.012

Reghioui, M., Prins, C., & Labadi, N. (2007). GRASP with Path Relinking for the Capacitated Arc Routing Problem with Time Windows. In M. Giacobini (Ed.), *Applications of Evolutionary Computing* (Vol. 4448, pp. 722-731): Springer Berlin Heidelberg.

Riquelme-Rodríguez, J.-P., Langevin, A., & Gamache, M. (2014). Adaptive large neighborhood search for the periodic capacitated arc routing problem with inventory constraints. *Networks, 64*(2), 125-139. doi: 10.1002/net.21562

Ropke, S., & Pisinger, D. (2006a). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science, 40*(4), 455-472. doi: doi:10.1287/trsc.1050.0135

Ropke, S., & Pisinger, D. (2006b). A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research, 171*(3), 750-775. doi: http://dx.doi.org/10.1016/j.ejor.2004.09.004

Salazar-Aguilar, M. A., Langevin, A., & Laporte, G. (2012). Synchronized arc routing for snow plowing operations. *Computers & Operations Research, 39*(7), 1432-1440. doi: http://dx.doi.org/10.1016/j.cor.2011.08.014

Salazar-Aguilar, M. A., Langevin, A., & Laporte, G. (2013). The synchronized arc and node routing problem: Application to road marking. *Computers & Operations Research, 40*(7), 1708-1715. doi: http://dx.doi.org/10.1016/j.cor.2013.01.007

Tagmouti, M., Gendreau, M., & Potvin, J.-Y. (2007). Arc routing problems with time-dependent service costs. *European Journal of Operational Research, 181*(1), 30-39. doi: http://dx.doi.org/10.1016/j.ejor.2006.06.028

Tan, G., & Sun, J. (2011). An Integer Programming Approach for the Rural Postman Problem with Time Dependent Travel Times. In B. Fu & D.-Z. Du (Eds.), *Computing and Combinatorics* (Vol. 6842, pp. 414-431): Springer Berlin Heidelberg.

Wøhlk, S. (2005). *Contributions to arc routing.* University of Southern Denmark.

This article was submitted as:

Monroy-Licht, M, Amaya, C.A., & Langevin, A. ALNS for the rural postman problem with time windows. *Networks*. IN SUBMISSION

Preliminary results were presented at:

Monroy-Licht, M, Amaya, C.A., & Langevin, A. (2014) The rural postman problem with time windows. *IIE Annual Conference*. May 31-June 3, Montréal, Canada

Monroy-Licht, M, Amaya, C.A., & Langevin, A. (2014) Solution methods for the rural postman problem with time windows. *INFORMS annual meeting.* November 9-12, San Francisco, United State

CHAPTER 5 ARTICLE 3 : THE RESCHEDULING CAPACITATED ARC ROUTING PROBLEM

Marcela Monroy-Licht[1,2], Ciro Alberto Amaya[3], André Langevin[1,2], Louis-Martin Rousseau[1,2]

[1]Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal, Canada
[2]Centre de recherche sur les réseaux d'entreprises, la logistique et le transport (CIRRELT), Montréal, Canada

[3]Departamento de Ingeniería Industrial, Universidad de Los Andes, Bogotá, Colombia

Abstract

In this paper, the *rescheduling capacitated arc routing problem* (RCARP) is introduced. This is a dynamic routing and scheduling problem that considers adjustments to an initial itinerary when one or more vehicle failures occur during the execution stage and the original plan must be modified. We minimize the operational and schedule disruption costs. Formulations based on mixed integer programming are presented to compare different policies in the rerouting phase. A solution strategy is developed when both costs are evaluated and it is necessary to find a solution quickly. Computational tests on a large set of instances compare the performance of the formulations for different decision-maker policies.

Keywords: Rescheduling, Disruption schedule costs, Mixed-integer programming, CARP

5.1 Introduction

Winter maintenance operations such as snow plowing or salt spreading can be modeled using a *capacitated arc routing problem* (CARP) formulation. Indeed, the solution of this problem produces a schedule for the drivers, i.e., a sequence of road segments to visit. Once each vehicle starts its schedule, it should follow the planned itinerary if no unexpected event happens.

Various disruptions may occur during the execution of the plan; they include traffic jams, vehicle breakdowns, and traffic accidents. It may also be necessary to include new services. These changes may make the original plan no longer optimal or even feasible. When an important disruption occurs, the dispatcher must adjust the available resources if possible or

arrange overtime if the tasks cannot be postponed to the next day. Usually, all winter maintenance routing operations must be completed because safety is the priority (Campbell et al., 2014). Thus, if the vehicles fail or are involved in accidents, the plan must be adjusted based on the current state of the system.

When a disruption occurs, the routes should be quickly adjusted in real time to minimize negative effects. The vehicles are often equipped with communication technologies that provide real-time information, so up-to-date data is available to the rescheduling algorithms.

There are different costs to consider in the rerouting phase: operational costs, delay costs, and disruption costs. If only the operational and delay costs are minimized, the initial schedule could be considerably altered (Li et al., 2009). In some cases it is necessary to keep the changes in schedule low, since drivers may not be familiar with the new itineraries. Deviation from the original plan may cause service delay or drivers' overtime work.

Some research has been done on dynamic arc routing problems. Moreira et al. (2007) studied a cutting path determination problem, modeled as an undirected *rural postman problem* (RPP). The dynamic essence arises because the associated graph is allowed to change during the solution procedure: it is possible to incorporate new edges, in the context of the minimization of the overall distance covered. Tagmouti et al. (2011) presented a dynamic version of the CARP *with time-dependent service costs*; the optimal service-time intervals change due to weather updates. Yazici et al. (2014) studied a dynamic CARP for the *multi-robot sensor-based coverage problem*. When the robots encounter unexpected obstacles due to partially unknown nature of the environment, a rapid re-planning is necessary. Finally, Liu (2014) explored the dynamic CARP, considering the availability of the vehicles, the accessibility of the roads, added and deleted tasks and demands, and traffic congestion. A memetic algorithm with a new split scheme is proposed to minimize the total distance traveled.

For vehicle routing with vehicle breakdowns, Li et al. (2004) introduced the *vehicle rescheduling problem*. They studied vehicle breakdowns in public transit systems. They initially modeled the problem as a sequence of static vehicle scheduling problems and solved it using an auction-based algorithm (Li et al., 2004). They also developed a decision support system for an application involving rescheduling trucks for solid waste collection (Li et al., 2007). These series of Li et al. (2004, 2007a, and 2007b) are based on the *single depot vehicle scheduling problem*,

which is the problem of assigning vehicles to a set of predetermined trips with fixed start and end times. The authors assumed that scheduled trips, other than the disrupted trip, must not be delayed. They minimized the operational and delay costs. Li et al. (2009) expanded on their earlier work by simultaneously considering scheduling disruptions and trip delays in the rerouting problem with time windows. The rerouting problem was formulated as a *set covering problem*. They proposed a Lagrangian-relaxation-based insertion heuristic that includes an insertion-based algorithm to obtain a feasible solution for the primal problem. Some tests were executed using Solomon (1987) instances, and the results showed a considerable cost saving over the manual solution approach.

Mu et al. (2011) studied the *vehicle routing problem with vehicle breakdowns*. They proposed a mathematical formulation and two tabu search algorithms that minimize the operational costs but not the deviation from the original plan. They assume that one extra vehicle is available at the depot and could be used. They minimized the number of vehicles used and the total distance traveled after the disruption to complete the deliveries. They tested the algorithm on a set of problems generated based on standard vehicle-routing benchmark instances.

Dealing with disruptions is a complicated decision-making process; therefore a decision-support system with effective algorithms is invaluable. This paper presents the case where disruption is caused by vehicle failure during the execution of a CARP itinerary. We consider the operational and disruption costs. To the best of our knowledge, this is the first time that schedule disruption costs have been considered in dynamic arc routing problems.

This paper is organized as follows. The problem is defined in Section 5.2. An exploration of indicators of disruptions costs is presented in Section 5.3. Section 5.4 proposes a mixed integer programming formulation and discusses different policies. Section 5.5 presents the computational experiments and discusses a solution strategy that aims to simultaneously minimize the operational and disruption costs. At the end, Section 5.6 provides concluding remarks.

5.2 Problem definition

We now formally define the RCARP.

An initial schedule is given for a network defined on a graph $G = (V, E)$, where V is the set of vertices, E is the set of edges, and the subset of required edges is denoted R. Let K be a set of

vehicles of equal capacities that leave the depot, located at the vertex v_0, and follow an assigned itinerary (a sequence of edges to visit). We assume that all the vehicles start the working day at the same time, t_0. Let $B \subset K$ be the set of vehicle breakdowns, with breakdown time, BT, equal to $t_b > t_0$. While minor vehicle failures can be repaired quickly, serious failures require longer repair times, and in the latter case the dispatcher must adjust the schedule for other vehicles. The RCARP defined in this paper considers:

i. Vehicle breakdowns: We allow more than one vehicle breakdown at the same time. However, it is unlikely that more than one vehicle will fail on the same day.

ii. Active vehicles: If at BT any vehicle in K has already finished its planned itinerary, it is no longer available. We do not have extra vehicles at the depot. The set of active vehicles K_R consists of those vehicles still in operation at t_b.

iii. Multiple starting points: During the execution of the initial plan, the vehicles are at different locations. An active vehicle cannot be diverted until it has finished the service in progress on a given edge. Therefore, in the rerouting plan, the starting point for active vehicle k, v_k^0, is the end vertex of its current edge. Note that all active vehicles must end at the depot, vertex v_0.

iv. Requests: The set of required edges $R_R = R_o \cup R_b$ for the rerouting consists of the currently unserved edges of the active vehicles (denoted R_o) and the unserved edges of the failed vehicles (denoted R_b). If at the BT a vehicle k in B was serving an edge e, that edge is included in R_b.

v. No cancellations: All the requests defined in the initial plan must be satisfied; we do not allow cancellations. At the BT, the capacity of each vehicle is the capacity remaining after the completion of the service in progress. However, if this is not given explicitly, we relax the capacity constraints, and therefore the active vehicles may exceed their capacities in order to satisfy all the requests.

The RCARP seeks to find a new schedule in which routes start at multiple points and finish at the depot, all the requests are served, and a cost function is minimized.

5.3 Measures of disruption cost

Several works have agreed that traverse distance or time spent traversing the edges can be associated to an operational cost measure in routing problems. However, the definition of disruption cost remains largely unexplored. This section summarises some indicators that enable perturbation cost to be measured in routing problems.

Kelleher and Cavichiollo (1999) studied disruption in a manufacturing system. Disruption is defined as differences in the allocation, time, or resources of common operations in two schedules. They define a disruption function. Inserting a new operation or moving an operation to the right does not change the function value, and moving an operation to the left or to another resource increases the function value. The total disruption is the sum of the disruption for each operation. This approach was extended to vehicle routing by Rhalibi and Kelleher (2003), but no results were reported.

The disruption metric must be able to measure the similarity of two schedules. If minimal disruption is desirable, the new schedule must be as much as possible similar to the original one. If a CARP solution is represented as ordered sequences of requests (order in which each vehicle visits the assigned requests), the similarity of two solutions could be defined as the distance between the corresponding permutations of requests. We now summarize some metrics studied in optimization problems to determine the distance between order-based encoding solutions.

Let $s = (s_1, s_2, \dots, s_n)$ and $t = (t_1, t_2, \dots, t_n)$ be two permutations of elements. In our case the elements in s and t represent indices of required edges. Now we can define the following metrics.

5.3.1 Edit distance

The *edit distance* was first used to calculate the distance between strings composed of characters from a finite alphabet Σ. Λ is the null-character, signifying the absence of a character. Given two strings s and t on alphabet Σ, the edit distance $d_e(s, t)$ is the minimum-weight sum of *edit operations* that transforms s into t. One of the simplest sets of edit operations is that defined by Levenshtein (1966):

- Insertion of a single character: If $a = u$, then inserting the character v produces $a = uv$.
- Deletion of a single character: If $a = uv$, then deleting the character v changes $a = uv$ to $a = u$.

– Substitution of a single character: If $a = uv$, then substituting v for $w \neq v$ produces $a = uw$.

Sörensen (2006) adapted the edit distance to *vehicle routing problems*. Verbally, the *edit distance* calculates the minimal number of edit operations, weighted with 1, required to transform the first solution into the second.

5.3.2 Exact match

The *exact match* was introduced by Ronald (1998). It is an n-ary extension of the Hamming distance to an alphabet or set of numbers. The distance function is the number of matched values. Given s and t, the distance $d_m(s,t)$ is

$$d_m(s,t) = \sum_{i=1}^{n} m_{s,t,i} \tag{5.1}$$

where $m_{s,t,i} = \begin{cases} 1 \; if \; s_i = t_i \\ 0 \quad otherwise. \end{cases}$

5.3.3 R-type distance

This metric was proposed by Martí et al. (2005), and it is useful when the relative ordering of the items in the sequences is relevant. Given the permutations s and t, the distance $d_R(s,t)$ is defined to be the number of times each item does not immediately follow its successor in the other solution. Formally,

$$d_R(s,t) = \sum_{i=1}^{n-1} z_i \tag{5.2}$$

where $z_i = \begin{cases} 1 \; if \; \nexists j : s_i = t_j \; and \; s_{i+1} = t_{j+1} \\ 0 \quad otherwise. \end{cases}$

If we extend the codification of each sequence, by substituting each index of a required edge by the indices of its two incident nodes, the *R-type distance* represents the number of common edges plus the number of times consecutive requests appear in both sequences.

5.3.4 Longest sequence

We define this metric as the length of the longest common sequence in permutations s and t.

Measure distances are defined for permutations of the same size, but this can be generalized by adding null-characters to the shorter permutation. An adaptation of measure distances to routing solutions can be derived from Sörensen (2006).

5.4 Formulations

The objective function of an initial schedule in arc routing usually minimizes the total operational cost. When a disruption occurs, there may be additional costs, such as delay and deviation costs from the original plan. We now present a mixed integer formulation for the RCARP. To evaluate different rerouting policies we define different objective functions based on the operational and disruption cost, each of them being more or less advantageous.

Most of the notation was defined in Section 5.2. The formulation is defined on a graph G. Each edge in E is replaced by two arcs (i,j) and (j,i) to identify the traversal direction. Let A be the set of arcs (i,j) and (j,i) such that $(i,j) \in E$; let A_R be the set of arcs (i,j) and (j,i) such that $(i,j) \in R_R$; and let P be the set of pairs (i,j), (j,i), such that (i,j) and (j,i) represent the same required edge. To simplify the notation an arc (i,j) is denoted a. Finally, let $\delta^+(v)$ be the set of arcs with end vertex v, $\delta^-(v)$ the set of arcs with initial vertex v, and $\delta(S)$ the set of arcs with both vertices in the set of vertices S.

Let c_a be the traversal cost of arc a, d_a be the demand of arc a, and cap_k be the remaining capacity of vehicle k. The decision variables are

$x_{a,k} = 1$ if arc a is serviced by vehicle k and 0 otherwise;

$y_{a,k} =$ number of times arc a is traversed by vehicle k.

5.4.1 Objective 1 (O1): Minimizing the total distance traveled

In this case the policy seeks a new schedule that takes into account only the operational cost.

$$\text{minimize } Z = \sum_{k \in K_R} \sum_{a \in A} c_a y_{a,k} \tag{5.3}$$

s.t.:

$$\sum_{k\in K_R}(x_{a,k} + x_{a\prime k}) = 1 \quad \forall (a, a') \in P \tag{5.4}$$

$$\sum_{a\in\delta^+(v)}(x_{a,k} + y_{a,k}) = \sum_{a\in\delta^-(v)}(x_{a,k} + y_{a,k}) \;\; \forall\, v \in V | v \neq v_k^0, v \neq v_0, v = v_k^0 = v_0, \;\; k \in K_R \tag{5.5}$$

$$\sum_{a\in\delta^+(v_k^0)}(x_{a,k} + y_{a,k}) - \sum_{a\in\delta^-(v_k^0)}(x_{a,k} + y_{a,k}) \quad = 1 \quad \forall \;\; k \in K_R \tag{5.6}$$

$$\sum_{a\in\delta^+(v_0)}(x_{a,k} + y_{a,k}) - \sum_{a\in\delta^-(v_0)}(x_{a,k} + y_{a,k}) \quad = -1 \quad \forall \;\; k \in K_R \tag{5.7}$$

$$\sum_{a\in\delta(s)}(x_{a,k} + y_{a,k}) \leq M_S \sum_{a\in\delta^-(s)}(x_{a,k} + y_{a,k}) \quad \forall\, s \subseteq S, \;\; k \in K_R \tag{5.8}$$

$$x_{a,k} \in \{1,0\} \; \forall \; a \in A_R, k \; \in K_R \tag{5.9}$$

$$y_{a,k} \in \; \mathbb{Z}^+ \; \forall \; a \in A, k \; \in K_R \tag{5.10}$$

We look for a set of trips that minimizes the total traversal cost (5.3). Constraints (5.4) specify that all required edges must be serviced. Constraints (5.5)–(5.7) ensure flow conservation for each vertex. The connectivity constraints are specified by (5.8), where M_s is equal to twice the number of variables in $\delta(s)$. Finally, decision variables are defined by (5.9) and (5.10).

This problem is NP-hard because it is an extension of the RPP, which is NP-hard if there is more than one connected component (Lenstra and Kan, 1976). Indeed, suppose we add an artificial arc of cost zero from the depot v_0 to each initial vertex v_k^0. The problem can be formulated as a RPP with an additional constraint: the minimum tour must return to the depot at least $|K_R|$ times.

5.4.2 Objective 2 (O2): Minimizing the total distance traveled and considering capacity

This formulation is similar to the previous one, but in this policy one and only one vehicle is allowed to exceed its capacity while the operational cost is minimized. We add the variables $w_k = 1$ if vehicle k exceeds the remaining capacity cap_k, and $w_k = 0$ otherwise. Let the constant C be the initial vehicle capacity. The MIP-formulation includes the objective function (5.3) and constraints (5.4)–(5.10). The following constraints are added:

$$\sum_{a\in A_R} d_a x_{a,k} \leq cap_k + Cw_k \quad \forall k \in K_R \tag{5.11}$$

$$\sum_{k\in K_R} w_k \leq 1 \tag{5.12}$$

$$w_k \in \{1,0\} \forall \ k \ \in K_R \tag{5.13}$$

Constraints (5.11) and (5.12) specify that only one vehicle can exceed its remaining capacity. The w_k variables are defined by (5.13). This problem is NP-hard because it is an extension of the previous one.

5.4.3 Objective 3 (O3): Minimizing disruption cost

This policy looks for a new schedule as close as possible to the initial one. The formulation in this case seeks to minimize the changes to the baseline solution.

Although several indicators of disruption were discussed previously, we propose a metric that penalizes the insertions in a different route from the baseline solution; we believe that this allows getting a solution very similar to the initial one. The metric that measures the difference between the baseline solution and the rescheduling solution is *the number of edges moved to a different route*. This gives us information about the number of edges served by a different vehicle and the number of edges that must be added to the active routes.

We assume that the baseline solution is defined by $\bar{X}$ and $\bar{Y}$, where $\bar{X}$ and $\bar{Y}$ are vectors of values of the decision variables $\bar{x}_{a,k}$ and $\bar{y}_{a,k}$ determined in a similar way to $x_{a,k}$ and $y_{a,k}$. The values of $\bar{x}_{a,k}$ are 1 or 0, indicating whether or not arc a is served by vehicle k. Although the variables $y_{a,k}$ (and therefore $\bar{y}_{a,k}$) are restricted to be integer (10), the upper bound on these variables is 1 in a CARP formulation where the problem is effectively converted into a directed problem (Eglese and Letchford, 2000). We assume that the baseline solution satisfies this.

Given a baseline solution $\bar{X}, \bar{Y}$, let M_X be a coefficient matrix with component $m_{X\,a,k}$ equal to 0 if $\bar{x}_{a,k}$ is equal to 1 in $\bar{X}$, and equal to 1 otherwise. An analogous matrix M_Y can be defined for $\bar{y}_{a,k}$ in $\bar{Y}$. Then, the *number of edges moved to a different route (DE)* is given by

$$DE = \sum_{a\in A_R} \sum_{k\in K_R} m_{X\,a,k} x_{a,k} + \sum_{a\in A} \sum_{k\in K_R} m_{Y\,a,k} y_{a,k} \tag{5.14}$$

Let DE be the measure of disruption cost. We can now formulate a RCARP that seeks to minimize the disruption cost. The MIP-formulation is

$$\min Z = \sum_{a \in A_R} \sum_{k \in K_R} m_{X_{a,k}} x_{a,k} + \sum_{a \in A} \sum_{k \in K_R} m_{Y_{a,k}} y_{a,k} \quad (5.15)$$

The constraints are given by (4)–(10).

This problem is NP-hard because it can be expressed as a min-cost allocation problem. The cost of allocating the required edges to the remaining routes is equal to the number of edges needed to join them to the routes.

5.4.4 Objective 4 (O4): Minimizing operational and disruption cost

After a disruption, minimizing the operational costs and the disruption costs are objectives that often are in conflict, therefore it is necessary to find a trade-off to solve the problem. This policy pretends to minimize both costs at the same time.

If only traveled cost is considered, the baseline solution might be significantly altered. The dispatcher must look for a good solution in terms of operational cost trying to keep the changes in the re-scheduling plan low; otherwise it may be too expensive in terms of disruption costs.

This formulation seeks to minimize a cost that is weighted by the value of the baseline decision variables. The model is

$$\min Z = \sum_{a \in A_R} \sum_{k \in K_R} m_{X_{a,k}} c_a x_{a,k} + \sum_{a \in A} \sum_{k \in K_R} y_{a,k} \quad (5.16)$$

The constraints considered are defined by (5.4)–(5.10).

5.5 Results

The main objective of our computational experiments is to compare different rerouting policies. We present tests on a set of generated problems and on a set of larger problems derived from a real network. We then present a solution strategy which reduces the computational time and we discuss it for a specific case. Finally, we calculate different disruption metrics for our solutions.

5.5.1 Test set and baseline solution

We used some benchmarks instances for the CARP to generate instances for the RCARP. Initially we selected a group of undirected CARP benchmarks from the val set (Benavent et al. 1992), the egl set (Golden et al. 1983), the egl-large set (Brandão and Eglese, 2008), and the BMCV set (Beullens et al. 2003). The picked group comprised instances with at least 32 required edges, at least 3 vehicles, and at least 10 required edges per vehicle. The test set includes: 21 val, 4 egl, 56 BMCV, and 6 egl-larger instances, for a total of 87 instances.

We then determined a baseline solution for each instance. We obtained this solution from the solution of the CARP. Although CARP solutions are given by the above authors, we did not use these solutions if the authors did not list the values of the decision variables. Some solutions were kindly provided to us by Bode (2014). In the other cases, we ran a heuristic to find a feasible solution for the CARP. We implemented a route-first split-second algorithm (Prins et al. 2014). We solved the routing phase using a version of the *adaptive large neighborhood search* for the RPP with time windows (Monroy-Licht el a. 2015), and we solved the split phase using dynamic programming. Finally, we defined the initial schedule for each vehicle: order in which each vehicle traverses its edges.

5.5.2 Comparison of policies

We now compare the formulations of Section 5.4. The tests are carried out for the situation where only one vehicle breaks down, since this is the most common case. However, the models allow multiple simultaneous failures.

For each instance, we simulated a vehicle failure by choosing the breakdown vehicle randomly. Three BT were set for the same vehicle: BT= 0.3 t_a, 0.5 t_a, and 0.7 t_a, where t_a is the average route length of the baseline solution. Each BT represents an earlier, middle, and later breakdown vehicle. Finally, we updated the information after each vehicle failure. In order to prevent confusion we name “problem” the result of simulating a vehicle failure at a specific BT for an instance.

We solved the 4 models (O1, O2, O3, and O4) for each problem. We iteratively added connectivity constraints (see inequality 5.8) when the graph is not connected in the relaxed

solution. We solved the mixed-integer linear problems using the standard parameters of CPLEX 12.4.0 implemented on a 2.38-GHz AMD 250.

In the analysis, we include only the problems with at least 10 requests, at least one request still unserved by the broken-down vehicle, at least 5 requests per active vehicle, and at least 2 active vehicles. In order to present an appropriate comparison, problems not solved to optimality within two hours of computational time for all the objectives are excluded from the analysis. Table 5.1 presents the description of the problems solved for each BT value. It gives the number of problems solved to optimality; the minimum, maximum, and average number of initial vehicles ("Vehicles"); the minimum, maximum, and average $|R|$ and $|R_R|$ ("Request$_0$" and "Requests$_r$"); and the average number of active vehicles ("Vehicles$_r$").

Table 5.1: Characteristic of problems

Feature	$\boldsymbol{BT = 0.3\ t_a}$	$\boldsymbol{BT = 0.5\ t_a}$	$\boldsymbol{BT = 0.7\ t_a}$
No. Problems	63	66	45
Vehicles	$3 - 8 - 4.40$	$3 - 8 - 4.53$	$3 - 6 - 4.42$
Rrequest$_0$	$32 - 107 - 65.33$	$34 - 121 - 67.81$	$39 - 121 - 71.02$
Requests$_r$	$20 - 76 - 44.92$	$16 - 56 - 33.15$	$11 - 42 - 22.44$
Vehicles$_r$	3.30	3.29	2.76

BT : breakdown time

Table 5.2 summarizes the comparison of objectives. This table gives the average values of extra travel cost with respect to the baseline solution as a percentage (% extra cost); the number of services moved to a different route as a percentage of the total requests in the updated graph (% diff. services); the number of edges moved to a different route as a percentage of the number of edges in the graph (% diff. edges); the number of vehicles that exceed their capacities (veh_over); the maximum percentage by which the capacity is exceeded (% exc. cap); and the computational time in seconds (time).

Table 5.2 shows that minimizing the travel and disruption costs are conflicting objectives. While O1 obtains 1, 0.8, and 0.68 times the value of % diff. edges reported by O3 for each BT, O3 increases the extra cost by 1.8, 1.04, and 0.64 times the value of extra cost reported by O1. Note that if the vehicle breakdown occurs earlier the negative impact on the extra cost is

consistently lower. This behavior can be explained as follows. On an earlier BT there are more available resources close to the failure zone, contrary to a later BT, the resources would be far away from the failure zone and it might be more expensive to move the active vehicles.

Table 5.2: Comparison of objectives

Statistics	***BT***	**O1**	**O2**	**O3**	**O4**
% extra cost	0.3	2.39	2.58	6.71	3.62
	0.5	4.51	4.88	9.23	5.67
	0.7	7.74	7.99	12.74	8.19
% diff. Services	0.3	67.22	67.70	26.98	27.85
	0.5	58.33	54.37	25.61	26.26
	0.7	55.31	55.00	25.88	26.20
% diff. Edges	0.3	58.16	58.82	28.95	30.91
	0.5	41.61	39.57	23.05	24.91
	0.7	27.46	27.63	16.31	17.75
veh_over	0.3	1.62	0.98	1.38	1.46
	0.5	1.59	0.94	1.23	1.29
	0.7	1.36	0.91	1.13	1.07
% exc. Cap	0.3	89.38	101.46	72.78	74.82
	0.5	76.55	91.70	71.62	70.30
	0.7	93.62	103.83	75.59	80.85
Time	0.3	355.36	696.85	9.50	19.07
	0.5	130.39	153.31	5.86	8.71
	0.7	76.99	103.53	23.78	44.85

O1 - Minimizing the total distance traveled
O2 - Minimizing the total distance traveled and considering capacity
O3 - Minimizing disruption cost
O4 - Minimizing operational and disruption cost

The capacity constraints complicate the problem, and therefore O2 requires more computational time to find optimal solutions. Including the disruption cost in the objective function (O3 and O4) seems to provide solutions where the veh_over value is not significantly different from the value for O2. In addition, the computational time for O3 and O4 is much lower than that for O2, and O3 and O4 give the best results for % exc. cap.

Figure 5.1 shows the values of % extra cost and % diff. edges for all the models and all the problems, grouped by *BT*. As one can observe, Figure 5.1a includes some negative values; in these cases the capacity constraints of the baseline problem makes more expensive the initial plan in terms of travel costs.

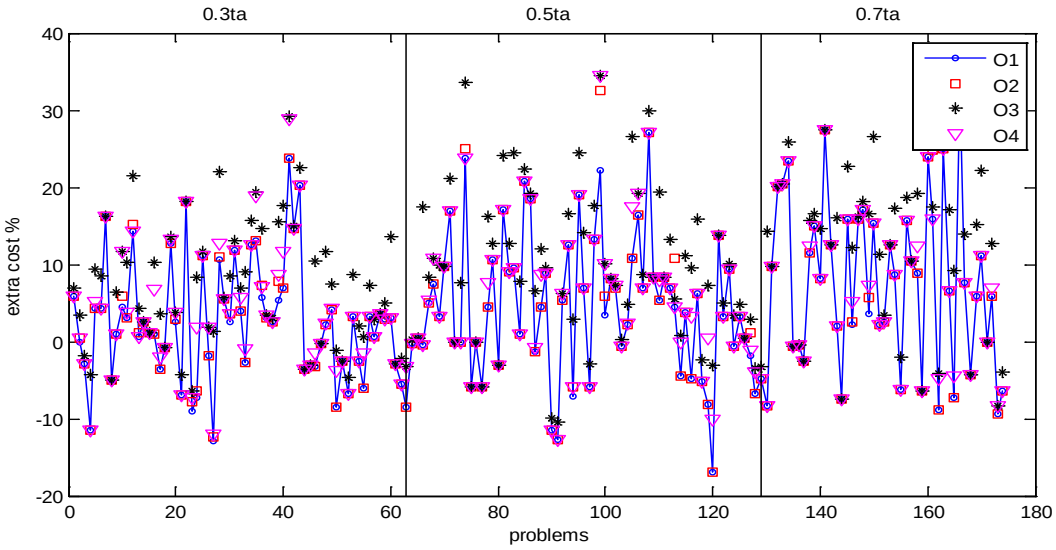


Figure 5.1a: % extra cost

The line connects the minimum values.

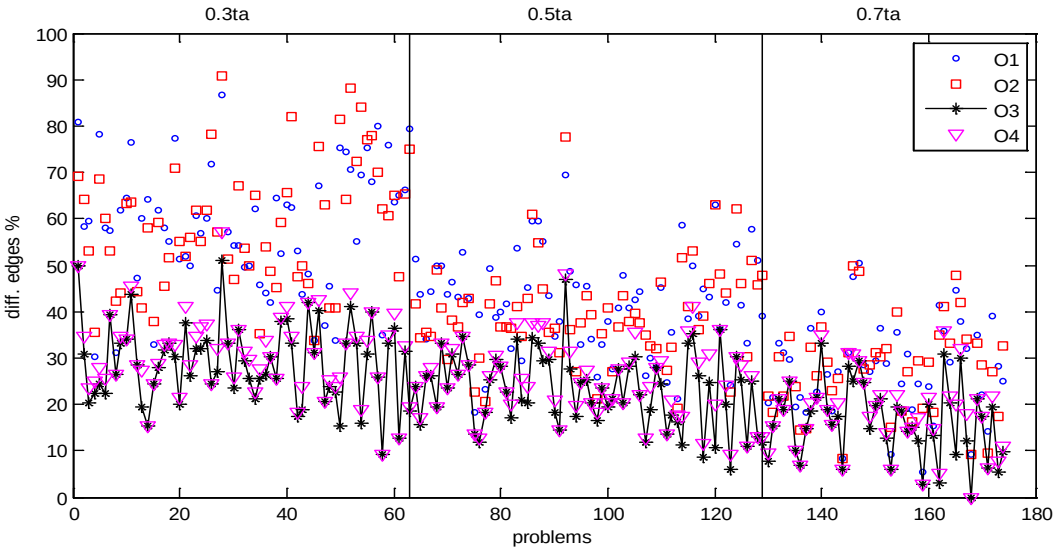


Figure 5.1b: Diff. edges

The line connects the minimum values.

Figure 5.1: Comparison of travel and disruption costs

In general, O4, which includes both travel and disruption costs, seems to be a good choice. It increases the minimum possible cost by factors of 0.51, 0.25, and 0.05 (compared to the O1

results), and it increases the minimum disruption cost by factors of 0.06, 0.08, and 0.08 (compared to the O3 results) for each subset of problems defined by a specific BT.

5.5.3 Larger networks

To test the performance of the formulations on larger networks we generated four undirected CARP instances based on the real network of the Eastern Townships an administrative region in the province of Quebec. This network has 140 nodes and 187 edges, all of them required. We set the number of vehicles to 9, 10, 11, and 12 and the capacities to 224, 196, 173, and 157. The baseline solutions of these instances were determined by the route-first split-second algorithm presented previously. For each instance we defined two tests, each with a different failed vehicle, and we again simulated vehicle failures at $0.3ta$, $0.5ta$, and $0.7ta$.

We solved O1, O2, O3, and O4 for each test. Table 5.3 shows for each objective and for each test the travel cost (cost), the number of edges moved to a different route (DE), the number of vehicles that exceeded the remaining capacity (v_o) and the computational time in seconds (time). The name of each test (test) is the number of initial vehicles followed by the index of the failed vehicle. Column $|R_r|$ is the number of requests at BT, $|NR_r|$ is the number of non-required edges at BT, and $|K_R|$ is the number of active vehicles. A "-" indicates that no optimal solution could be found within 75 minutes.

As noted from Table 5.3, O1 and O2 fail to solve the larger problems. O3 solves to optimality all cases, although the computational time can be high in some tests. O4 is competitive in terms of the instances solved, but its computational time is sometimes high. When O2 finds an optimal solution, the travel cost is the same as that reported by O1. The O2 solutions may be preferable because fewer edges are moved to different routes and at most one vehicle exceeds its capacity.

5.5.4 Solution strategy

Several alternative good solutions may better support a dispatcher when operational and disruption costs are considered. We present a solution strategy that aims at solving larger networks, giving quick response and thus short computation time and generating several alternative solutions for the decision-maker to choose from.

While minimizing the deviation from the original plan is an objective in the rescheduling, benefit might be taken from the original solution. Additionally, it will be quicker to find a new solution if we do not change all the initial solution.

Table 5.3: Comparison of results for larger networks

BT	test	$\|R_r\|$	$\|NR_r\|$	$\|K_R\|$	O1				O2				O3				O4			
					cost	DE	v_o	time	cost	DE	v_o	time	cost	DE	v_o	time	cost	DE	veh_o	time
0.3ta	9_1	134	53	8	-	-	-	-	-	-	-	-	25557	26	2	3600.8	-	-	-	-
	9_3	134	53	8	-	-	-	-	-	-	-	-	24603	29	1	454.6	22992	33	1	390.9
	10_4	139	48	8	-	-	-	-	-	-	-	-	27108	23	1	1419.1	-	-	-	-
	10_8	139	48	8	-	-	-	-	-	-	-	-	26021	28	1	79.7	24950	41	1	183.7
	11_3	137	50	9	-	-	-	-	-	-	-	-	27518	20	1	1731.7	25129	61	0	3621.9
	11_7	137	50	9	-	-	-	-	-	-	-	-	26952	22	1	140.5	25011	43	1	2162.6
	12_10	135	52	10	-	-	-	-	-	-	-	-	28613	20	1	81.4	25922	46	0	36.8
	12_2	135	52	10	-	-	-	-	-	-	-	-	28030	16	1	209.9	-	-	-	-
Average		136.3	50.8	8.8																
0.5ta	9_1	91	96	7	-	-	-	-	-	-	-	-	18151	20	3	36.1	17051	30	2	19.7
	9_3	91	96	7	15874	132	3	4194.0	15874	117	1	4164.4	17833	29	1	768.5	16389	31	1	195.9
	10_4	85	102	8	-	-	-	-	-	-	-	-	20700	14	1	243.2	-	-	-	-
	10_8	85	102	8	-	-	-	-	-	-	-	-	19606	24	1	93.9	-	-	-	-
	11_3	87	100	8	19135	128	3	4418.6	-	-	-	-	20817	15	1	329.9	19964	30	1	68.2
	11_7	87	100	8	-	-	-	-	-	-	-	-	20460	19	1	212.9	19538	42	1	591.5
	12_10	88	99	10	-	-	-	-	-	-	-	-	21953	16	0	1099.9	19959	32	0	6.1
	12_2	**88**	**99**	**10**	**18712**	**112**	**2**	**3841.2**	**18712**	**116**	**1**	**3661.9**	**20602**	**12**	**0**	**31.8**	**19614**	**24**	**1**	**92.7**
Average		87.8	99.3	8.3																
0.7ta	9_1	52	135	7	11137	55	2	417.1	11137	53	1	577.0	11662	7	1	2.0	11394	10	2	0.4
	9_3	52	135	7	10115	67	2	2127.0	10115	63	1	1697.6	11983	24	2	3600.1	10304	26	1	103.1
	10_4	44	143	8	12283	80	1	3146.9	12283	74	1	3221.1	13705	11	0	3619.9	12715	23	0	70.0
	10_8	44	143	8	11105	57	1	728.2	11105	57	1	759.1	12873	14	1	49.5	11262	27	1	40.6
	11_3	44	143	8	13343	57	1	273.6	13343	57	1	247.5	14605	9	0	2412.8	13654	25	0	3621.4
	11_7	44	143	8	12801	51	1	483.3	12801	47	0	630.7	13694	12	1	95.0	12851	29	1	232.9
	12_10	51	136	8	13703	41	1	128.1	13703	43	1	138.9	14770	11	0	3.9	13703	18	0	0.7
	12_2	51	136	8	12516	48	1	122.1	12516	44	1	125.0	13500	3	0	0.2	12796	6	1	0.2
Average		47.8	139.3	7.8																

Our strategy is based on two ideas. i) Reducing the size of the problem by setting some of the decision variables $x_{a,k}$ to their values in the original solution. On the one hand, requests that are far from requests in R_b will not be moved to another route. On the other hand, at the BT, possibly it is not necessary to reschedule all the routes: vehicles that are far from the requests in R_b should continue with their initial itineraries, whereas closer vehicles should be rescheduled. ii) Disruption costs reduction.

The strategy comprises two stages: the first one determines the set of fixed arcs and the set of vehicles to be rescheduled (see Algorithm 1). The second stage solves the RCARP with the new set of active vehicles and with additional constraints (see Algorithm 2).

Let A_R (defined in section 4) be partitioned into two sets A_o and A_b, where A_o is the set of arcs (i,j) and (j,i) such that $(i,j) \in R_o$, and A_b is the set of arcs (i,j) and (j,i) such that $(i,j) \in R_b$. Let per be the percentage of fixed arcs and $model$ be one of the formulations presented in Section 5.4. We now present Algorithms 1 and 2.

Algorithm 1: Selection of fixed arcs and vehicles to be rescheduled

Input: $A_o, A_b, per, K_R, \bar{X}$
Output: The set L contains the fixed arcs
The set K_{es} contains the vehicles to be rescheduled
Initialize the set of pairs $(arc, distance)$ to $Dis = \emptyset$
For a in A_o **do**
Find the nearest arc in A_b to arc a
$di_a \leftarrow$ Distance from a to the nearest arc in A_b
$Dis \leftarrow Dis \cup \{(a, di_a)\}$
End for
Initialize the set of arcs to $L_c = \emptyset$
While $n < (1 - per)|A_o|$ **do**
Choose the arc $\tilde{a}$ from Dis with minimum value $di_{\tilde{a}}$
$L_c \leftarrow L_c \cup \{\tilde{a}\}$
$Dis \leftarrow Dis - \{(a, di_a)\}$
End while
Initialize $K_{es} = K_R$
For k in K_R **do**
If $\nexists\ a \mid \bar{x}_{a,k} = 1, a \in L_c$ **then**
$K_{es} \leftarrow K_{es} - \{k\}$
End if
End for
$L \leftarrow R_o \setminus L_c$

Algorithm 2: Solution of the RCARP
Input: $model, K_{es}, L, \bar{X}$
Output: X, Y: solution of the RCARP
Load $model$
$K_R \leftarrow K_{es}$
For a in L and k in K_{es} **do**
 If $\bar{x}_{a,k} = 1$ **then**
  $x_{a,k} = \bar{x}_{a,k}$
 End if
End do
Solve $model$

Note that the solution strategy hardly improves the performance of model O3 regarding the disruption costs, but the computational time may be reduced.

We tested our solution strategy on the set of tests from larger networks. We tested three values for the parameter per (0.3, 0.5, and 0.7), and we tested Algorithm 2 for all four models (O1, O2, O3, and O4). The detailed results are reported in the annex 1. We now use an example to show how the solution strategy works. We selected one of the larger problems from Table 5.3 with optimal solution for all objectives. Indeed, we chose the problem: test 12_2, BT=0.5ta. Figure 5.2 illustrates the trade-offs between the travel cost and disruption cost (DE) of the solutions. Each strategy is labeled FIX-per-$model$, where per and $model$ are the values of the corresponding parameters. In Figure 5.2, the circles refer to models O1–O4. The markers "+" correspond to strategies with parameter $model$=O1, the markers "*" denote the strategies with parameter $model$ =O2, and the markers "x" indicate the strategies with parameter $model$ =O4. Note that the values reported by strategies FIX-0.3-O3, FIX-0.5-O3, and FIX-0.7-O3 are not included because they do not improve the current solution of model O3.

Figure 5.2 highlights the pareto-optimal solutions: O1, FIX-0.3-O1, O4, FIX-0.3-O4, FIX-0.5-O4, FIX-0.7-O4, and O3; their relative merits depend on the decision policy. Table 5.4 reports the computational times for all the strategies.

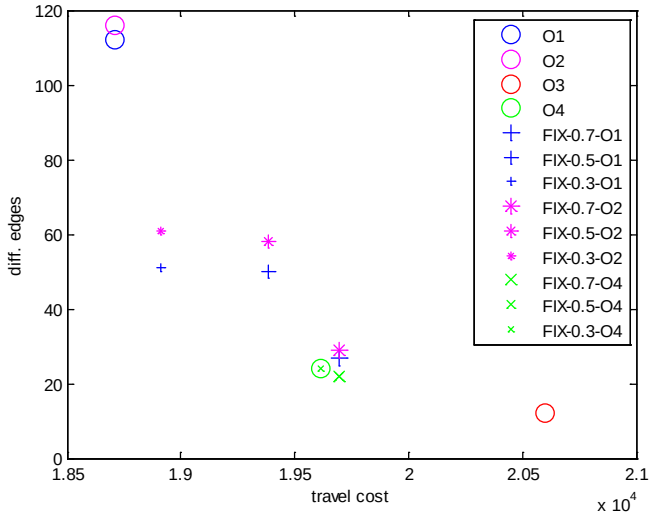


Figure 5.2: Travel and disruption costs (Problem 12_2, $BT = 0.5ta$)

Table 5.4: Computational times (Problem 12_2, $BT = 0.5ta$)

Strategy	FIX -0.7-O1	FIX -0.5-O1	FIX -0.3-O1	FIX -0.7-O2	FIX -0.5-O2	FIX -0.3-O2	FIX -0.7-O3	FIX -0.5-O3	FIX -0.3-O3	FIX -0.7-O4	FIX -0.5-O4	FIX -0.3-O4
Time (s)	13.5	40.5	301.2	10.6	53.3	332.0	1.0	2.2	8.9	2.9	4.2	22.1

We compare each solution with the previous solution that involved the same policy (see the bold row of Table 5.3). For policies O1 and O2, the solutions of the strategies considerably reduce the disruption costs. The solutions of the strategies that use $model$ =O1, and O2 are more diverse than those that use $model$ =O4, and the computational time is reduced by more than 90%. For $per = 0.5$ and $per = 0.7$ the solutions of the strategies that use $model$ =O4 are similar to the previous O4 solution; the solution for FIX-0.3-O4 is the same as that for O4. For O3 the solutions of the strategies are the same as the previous O3 solution, and for the strategies involving models O3 and O4 the computational time is reduced by more than 70%.

Figure 5.3 illustrates some solutions of the problem 12_2, BT =0.5ta. The baseline solution with 12 vehicles is presented in Figure 5.3a. The depot is located at the red node, and the requests of each vehicle are shown in different colors. Figure 5.3b concerns the requests in R_r. The dashed lines correspond to requests initially assigned to the failed vehicle, vehicle 2. At BT =0.5ta, there are eight active vehicles.

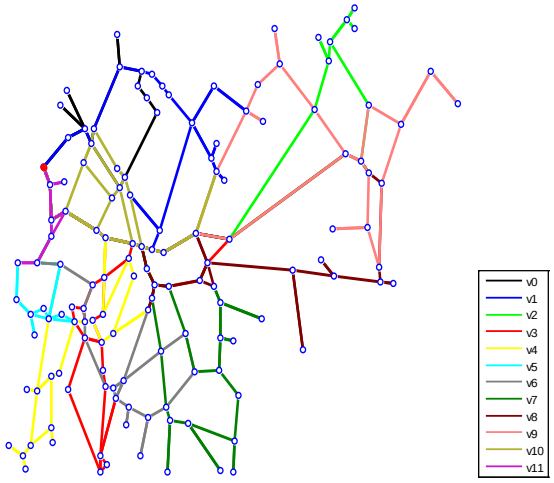


Figure 5.3a: CARP solution

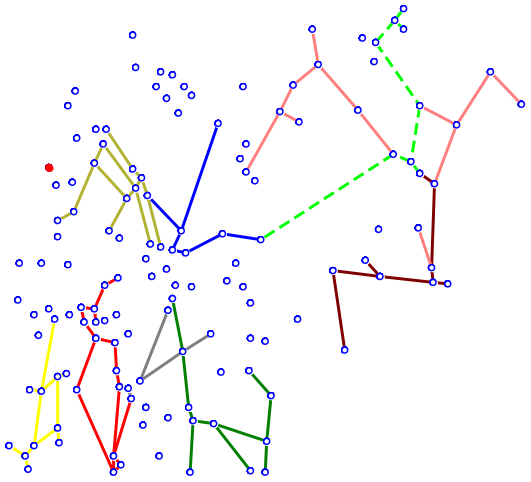


Figure 5.3b: Breakdown of vehicle 2 at $BT = 0.5ta$

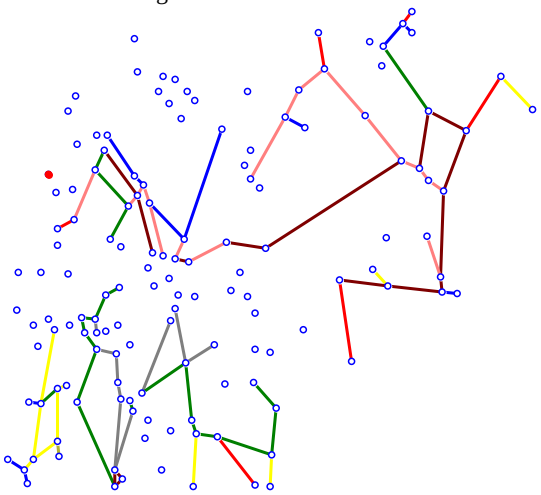


Figure 5.3c: RCARP solution for O1: minimize total travel cost

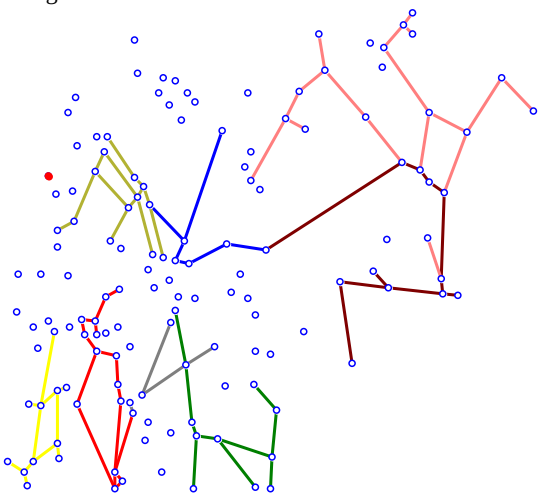


Figure 5.3d: RCARP solution for O3: minimize disruption cost

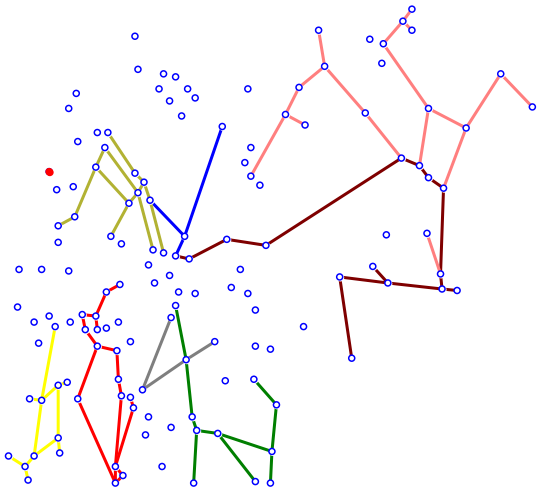


Figure 5.3e: RCARP solution for O4: minimize operational and disruption costs

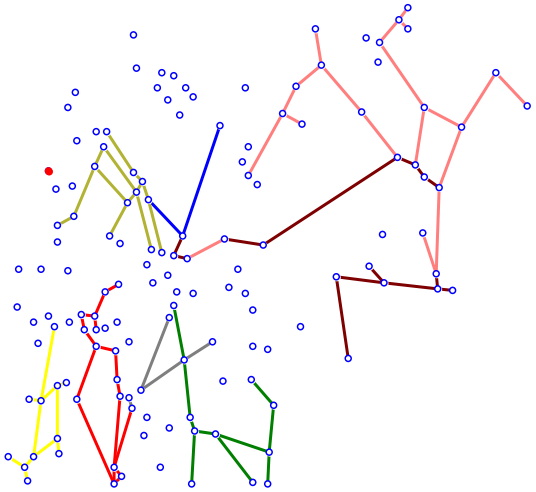


Figure 5.3f: RCARP solution with FIX-0.3-O1 for O1: minimize total travel cost

Figure 5.3: Example of different policies for problem 12_2, $BT = 0.5ta$

Figures 5.3c–5.3e illustrate the solutions for O1, O3, and O4 respectively. The O1 solution modifies the routes for all the active vehicles, and the geographical assignment of requests to vehicles is highly dispersed. The O3 solution modifies the routes for only two vehicles: the requests in R_b are reassigned to vehicles 8 and 9. The O4 solution modifies the routes for vehicles 1, 8, and 9: the requests in R_b are reassigned to vehicles 8 and 9, and some requests initially assigned to vehicle 1 are reassigned to vehicle 8. The solution FIX-0.3-O1, presented in Figure 5.3f, modifies the routes for vehicles 1, 8, and 9: the requests in R_b are reassigned to vehicles 8 and 9, and various other reassignments occur. This solution significantly improves the disruption cost of the O1 solution, reducing it by 54.5%, nevertheless the travel cost increases by only 1.7%.

5.6 Evaluation of metrics

We defined DE in Section 5.4.3 as a measure of the disruption cost. We compare this measure with other possible metrics of disruption costs by calculating their values in our solutions.

We present the values of edit distance, exact match, R-type distance, and longest sequence for twelve problems selected randomly from the subset of problems with BT=0.5ta. We selected this subset because it includes the largest problems. Figure 5.4 shows the values of the metrics for each solution of O1–O4.

In Figures 5.4a and 5.4c we plot the lower bound of edit distance and R-type distance, and in Figures 5.4b and 5.4d we plot the upper bound of exact match and longest sequence. Although O3 minimizes only the value of DE, its solutions are the closest on average to (but still relatively far from) the bounds of each metric. It is important to remember that the metrics are general distance measures for ordered sequences and studies of their effectiveness in the routing context would be interesting (this is beyond the scope of this work).

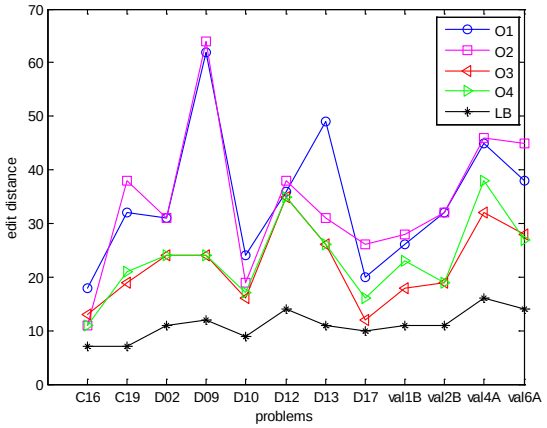


Figure 4a: Edit distance

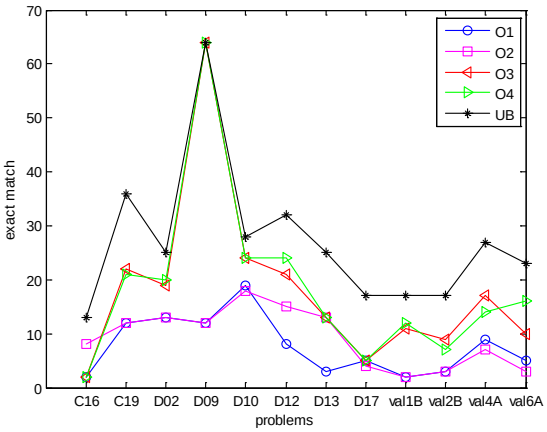


Figure 4b: Exact match

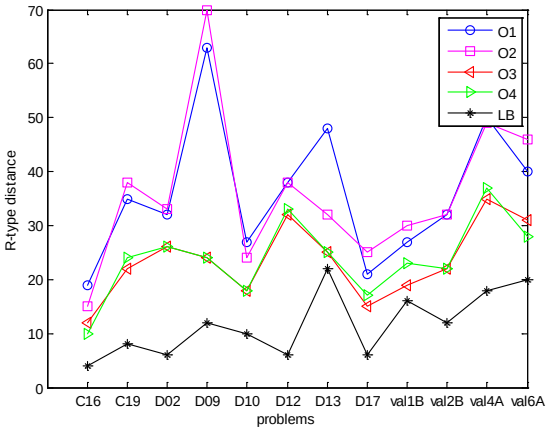


Figure 4c: R-type distance

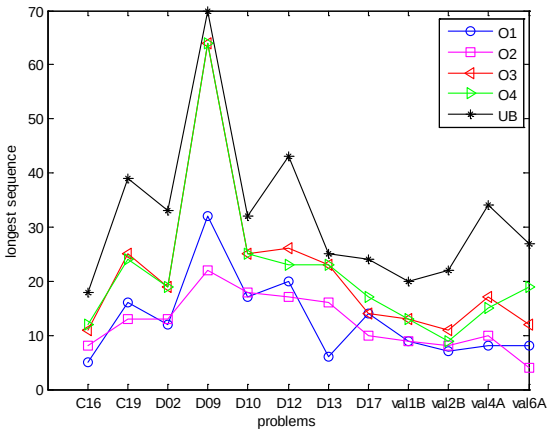


Figure 4d: Longest sequence

Figure 5.4: Disruption metrics

5.7 Conclusions

The RCARP is a challenging dynamic problem that aims to manage the disruptions caused by vehicle failures. We have studied for the first time this problem when disruption schedule costs are considered. A mixed-integer formulation is presented for different policies concerned to operational and schedule disruption costs.

We have observed that operational and disruption costs are objectives in conflict. The formulation that minimizes both costs (O4) works well, finding a trade-off between the conflicting goals and solving 18 of 24 large problems to optimality.

To provide a set of solutions when both costs are considered and to reduce the computational time, we have proposed a solution strategy. We tested it on the set of large instances. More diverse solutions were found when the strategy includes the minimization of the total travel cost

as objective function. We have explored some disruption metrics; but these metrics need to be adapted for rerouting problems.

Future work will consider other rerouting constraints such as route equilibrium or a maximum number of modifications. We will also explore the use of metaheuristics to get a larger set of pareto-solutions.

References

BENAVENT, E., CAMPOS, V., CORBERAN, A. & MOTA, E. 1992. The capacitated Chinese postman problem: Lower bounds. *Networks,* 22, 669-690.

BEULLENS, P., MUYLDERMANS, L., CATTRYSSE, D. & VAN OUDHEUSDEN, D. 2003. A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research,* 147, 629-643.

BODE, C. 2014. Private communication.

BRANDÃO, J. and EGLESE, R. 2008. A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research,* 35, 1112-1126.

CAMPBELL, J. F., LANGEVIN, A. & PERRIER, N. 2014. Advances in vehicle routing for snow plowing. *In:* ÁNGEL, C. & GILBERT, L. (eds.) *Arc Routing: Problems, Methods, and Applications.* SIAM.

EGLESE, R. & LETCHFORD, A. 2000. Polyhedral theory for arc routing problems. *In:* DROR, M. (ed.) *Arc Routing.* Springer US.

GOLDEN, B. L., DEARMON, J. S. & BAKER, E. K. 1983. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research,* 10, 47-59.

KELLEHER, G. & CAVICHIOLLO, P. 1999. Intelligent support of the rescheduling of complex manufacturing domains: An example application. *Second International Workshop on IMS.* Leuven, Belgium.

LENSTRA, J. K. & RINNOOY KAN, A. H. G. 1976. On general routing problems. *Networks,* 6, 273-280.

LEVENSHTEIN, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, 707-710.

LI, J.-Q., MIRCHANDANI, P. B. & BORENSTEIN, D. 2004. Parallel auction algorithm for bus rescheduling. Proceedings of the Ninth International Conference on Computer-Aided Scheduling of Public Transport. San Diego, California, USA.

LI, J.-Q., BORENSTEIN, D. & MIRCHANDANI, P. B. 2007a. A decision support system for the single-depot vehicle rescheduling problem. *Computers & Operations Research,* 34, 1008-1032.

LI, J.-Q., MIRCHANDANI, P. B. & BORENSTEIN, D. 2007b. The vehicle rescheduling problem: Model and algorithms. *Networks,* 50, 211-229.

LI, J.-Q., MIRCHANDANI, P. B. & BORENSTEIN, D. 2009a. A Lagrangian heuristic for the real-time vehicle rescheduling problem. *Transportation Research Part E: Logistics and Transportation Review,* 45, 419-433.

LI, J.-Q., MIRCHANDANI, P. B. & BORENSTEIN, D. 2009b. Real-time vehicle rerouting problems with time windows. *European Journal of Operational Research,* 194, 711-727.

LIU, M. 2014. A memetic algorithm with a new split scheme for solving dynamic capacitated arc routing problems. Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, September 16 2014, 595-602.

MARTÍ, R., LAGUNA, M. & CAMPOS, V. 2005. Scatter search vs. genetic algorithms. *In:* SHARDA, R., VOß, S., REGO, C. & ALIDAEE, B. (eds.) *Metaheuristic Optimization via Memory and Evolution.* Springer US.

MONROY-LICHT, M., AMAYA, C. & LANGEVIN, A. 2015. Adaptive large neighborhood search for the rural postman problem with time windows. Montreal: CIRRELT.

MOREIRA, L. M., OLIVEIRA, J. F., GOMES, A. M. & FERREIRA, J. S. 2007. Heuristics for a dynamic rural postman problem. *Computers & Operations Research,* 34, 3281-3294.

MU, Q., FU, Z., LYSGAARD, J. & EGLESE, R. 2011. Disruption management of the vehicle routing problem with vehicle breakdown. *J. Oper. Res. Soc.,* 62**,** 742-749.

PRINS, C., LACOMME, P. & PRODHON, C. 2014. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies,* 40, 179-200.

RHALIBI, A. & KELLEHER, G. 2003. An approach to dynamic vehicle routing, rescheduling and disruption metrics. IEEE International Conference on Systems, Man and Cybernetics, 5-8 Oct. 2003, 3613-3618 vol. 4.

RONALD, S. 1998. More distance functions for order-based encodings. Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, 4-9 May 1998, 558-563.

SOLOMON, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research,* 35, 254-265.

SÖRENSEN, K. 2006. Route stability in vehicle routing decisions: A bi-objective approach using metaheuristics. *Central European Journal of Operations Research,* 14, 193-207.

TAGMOUTI, M., GENDREAU, M. & POTVIN, J.-Y. 2011. A dynamic capacitated arc routing problem with time-dependent service costs. *Transportation Research Part C: Emerging Technologies,* 19, 20-28.

YAZICI, A., KIRLIK, G., PARLAKTUNA, O. & SIPAHIOGLU, A. 2014. A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints. *IEEE Transactions on Cybernetics,* 44, 305-314.

Annex 1. Solution strategy

Table 5.5: Strategy solution in larger instances

BT=0.3ta				***per*=0.7**			***per* =0.5**			***per* =0.3**		
test	***model***	$\|R_r\|$	$\|K_R\|$	**cost**	***DE***	**time**	**Cost**	***DE***	**time**	**cost**	***DE***	**time**
9_1	O1	134	8	25100	47	913.7	24047	93	1086.8	-	-	-
	O2	134	8	24983	50	928.4	-	-	-	-	-	-
	O3	134	8	25472	26	66.3	25557	26	907.7	25557	26	946.3
	O4	134	8	25170	27	131.4	24255	35	441.9	23790	53	938.5
9_3	O1	134	8	23782	74	936.9	-	-	-	-	-	-
	O2	134	8	23782	66	1068.7	-	-	-	-	-	-
	O3	134	8	23942	29	45.2	24431	29	119.3	24431	29	165.4
	O4	134	8	23862	31	10.4	23862	31	104.1	23314	33	194.5
10_4	O1	139	8	27024	79	933.6	26466	90	1130.9	-	-	-
	O2	139	8	26944	70	916.7	26466	95	915.0	25504	132	1377.6
	O3	139	8	28156	26	151.0	28180	26	117.6	27454	23	531.8
	O4	139	8	27137	40	409.5	27137	40	906.7	-	-	-
10_8	O1	139	8	25489	56	160.7	24552	101	1021.9	-	-	-
	O2	139	8	25489	60	309.7	24552	95	1041.2	-	-	-
	O3	139	8	26021	28	8.0	26021	28	49.3	26021	28	21.1
	O4	139	8	25613	30	6.5	25153	33	73.3	24950	41	196.0
11_3	O1	137	9	26500	58	1254.3	26189	96	1001.5	-	-	-
	O2	137	9	26500	63	933.0	-	-	-	-	-	-
	O3	137	9	27756	19	47.9	27932	24	103.1	28385	24	130.2
	O4	137	9	27068	29	45.6	26671	53	189.0	26289	48	754.9
11_7	O1	137	9	-	-	-	-	-	-	-	-	-
	O2	137	9	-	-	-	-	-	-	-	-	-
	O3	137	9	26952	22	44.1	26952	22	60.7	26952	22	61.6
	O4	137	9	26616	29	159.3	-	-	-	25011	43	605.8
12_10	O1	135	10	26337	58	262.4	-	-	-	-	-	-
	O2	135	10	26337	54	194.2	-	-	-	-	-	-
	O3	135	10	28613	20	2.6	28321	20	18.4	28613	20	8.1
	O4	135	10	26769	35	3.6	26097	43	5.5	26016	52	32.8
12_2	O1	135	10	-	-	-	-	-	-	-	-	-
	O2	135	10	26709	83	1260.9	-	-	-	-	-	-
	O3	135	10	29086	16	32.1	29086	16	69.4	29086	16	204.8
	O4	135	10	26863	29	16.0	-	-	-	25780	44	508.4

- Not solution found in less than 1400s.

Table 5.5: Strategy solution in larger instances

	BT=0.5ta			***per*=0.7**			***per* =0.5**			***per* =0.3**		
test	***model***	$\|R_r\|$	$\|K_R\|$	**cost**	***DE***	**time**	**cost**	***DE***	**time**	**cost**	***DE***	**time**
9_1	O1	91	7	18100	43	58.7	17263	59	176.7	16934	99	815.1
	O2	91	7	18100	38	43.2	17263	64	158.0	16934	93	716.0
	O3	91	7	18536	22	5.1	18151	20	30.7	18151	20	35.6
	O4	91	7	18100	28	21.5	17263	29	5.0	17051	30	4.5
9_3	O1	91	7	16670	44	58.5	16196	52	967.0	16196	73	1097.9
	O2	91	7	16670	40	120.1	16196	59	930.2	16196	68	1012.7
	O3	91	7	16880	29	35.2	17452	29	339.9	18364	29	320.4
	O4	91	7	16711	31	6.6	16711	31	79.3	16711	31	103.2
10_4	O1	85	8	19732	36	235.4	19551	58	687.5	18938	87	920.0
	O2	85	8	19732	33	451.0	19551	61	639.1	18938	97	904.6
	O3	85	8	20792	13	26.5	20792	13	13.2	20700	14	116.1
	O4	85	8	19877	25	27.2	19877	25	71.9	19263	47	86.7
10_8	O1	85	8	19580	53	588.0	19280	61	905.0	-	-	-
	O2	85	8	19580	53	827.6	19280	59	1030.9	-	-	-
	O3	85	8	20513	24	10.1	20513	24	28.3	20513	24	93.4
	O4	85	8	19601	24	32.0	19601	24	52.2	19063	36	210.3
11_3	O1	87	8	20520	30	68.0	19976	64	473.3	19536	91	943.3
	O2	87	8	20520	31	112.5	19976	63	652.3	19536	94	1144.8
	O3	87	8	21251	15	81.9	20817	14	89.1	21549	15	418.5
	O4	87	8	20537	20	4.4	20066	30	18.3	19964	30	32.7
11_7	O1	87	8	20155	37	268.2	19489	74	997.6	18842	105	1005.6
	O2	87	8	20155	48	317.2	19489	76	942.1	18842	93	934.0
	O3	87	8	20483	18	10.4	20460	19	82.8	20483	19	127.2
	O4	87	8	20155	23	22.8	19953	34	232.8	19538	42	289.2
12_10	O1	88	10	20120	30	1.7	20039	36	530.0	19958	82	937.2
	O2	88	10	20120	32	3.1	20039	43	561.6	19958	86	922.3
	O3	88	10	21953	16	64.7	21615	16	24.1	21615	16	260.2
	O4	88	10	20134	29	0.5	20053	31	1.6	20053	31	0.6
12_2	O1	88	10	19695	27	13.5	19386	50	40.5	18911	51	301.2
	O2	88	10	19695	29	10.6	19386	58	53.3	18911	61	332.0
	O3	88	10	20602	12	1.0	20602	12	2.2	20602	12	8.9
	O4	88	10	19695	22	2.9	19695	22	4.2	19614	24	22.1

- Not solution found in less than 1400s.

Table 5.5: Strategy solution in larger instances

	BT=0.7ta			***per*=0.7**			***per* =0.5**			***per* =0.3**		
test	***model***	$\|R_r\|$	$\|K_R\|$	**cost**	***DE***	**time**	**cost**	***DE***	**time**	**cost**	***DE***	**time**
9_1	O1	52	7	12051	18	0.5	11411	28	0.3	11278	33	6.4
	O2	52	7	12051	22	0.9	11411	32	1.6	11278	42	12.4
	O3	52	7	12161	10	0.6	11662	7	0.3	11662	7	0.2
	O4	52	7	12051	13	0.5	11467	9	0.1	11394	10	0.2
9_3	O1	52	7	10304	31	2.2	10231	37	10.5	10231	45	29.8
	O2	52	7	10304	30	2.7	10231	33	20.3	10231	48	88.4
	O3	52	7	11983	24	34.6	11983	24	102.5	11983	24	103.7
	O4	52	7	10304	26	0.3	10304	26	7.1	10304	27	6.1
10_4	O1	44	8	13260	19	253.0	12692	37	79.7	12692	49	400.7
	O2	44	8	13260	19	200.0	12692	38	91.3	12692	37	340.0
	O3	44	8	13664	12	890.6	13842	12	920.9	13863	10	904.3
	O4	44	8	13260	17	76.4	12738	25	7.2	12738	25	12.9
10_8	O1	44	8	11773	19	0.3	11773	19	20.9	11416	31	278.9
	O2	44	8	11773	19	0.7	11773	20	19.9	11416	31	225.6
	O3	44	8	12685	14	0.4	12873	14	4.7	11961	14	33.9
	O4	44	8	11773	14	0.1	11773	14	3.6	11416	21	8.0
11_3	O1	44	8	14268	20	2.2	13771	31	8.2	13521	38	20.0
	O2	44	8	14268	24	2.1	13771	34	9.2	13521	39	25.5
	O3	44	8	14605	9	323.4	14605	9	599.3	14605	9	527.7
	O4	44	8	14465	13	145.0	13867	22	137.8	13544	28	398.8
11_7	O1	44	8	13297	20	4.8	13297	25	13.9	12851	42	31.1
	O2	44	8	13297	23	5.7	13297	29	16.1	12851	34	47.9
	O3	44	8	13694	11	4.2	13694	12	9.2	13694	12	16.7
	O4	44	8	13297	17	1.9	13297	18	9.0	12851	29	15.6
12_10	O1	51	8	14458	21	0.7	13797	17	2.9	13797	25	20.0
	O2	51	8	14458	21	0.9	13797	19	5.5	13797	19	21.2
	O3	51	8	14770	11	0.4	14770	11	3.4	14770	11	0.8
	O4	51	8	14501	14	0.2	13797	17	0.6	13797	17	0.3
12_2	O1	51	8	12796	8	0.2	12588	29	1.4	12516	32	3.4
	O2	51	8	12796	6	0.1	12588	21	0.5	12516	28	6.3
	O3	51	8	13500	3	0.1	13500	3	0.1	13500	3	0.2
	O4	51	8	12796	6	0.1	12796	6	0.1	12796	6	0.1

- Not solution found in less than 1400s.

This article was submitted as:

Monroy-Licht, M., Amaya, C.A., Langevin, A., & Rousseau, L-M. The rescheduling capacitated arc routing problem. *International Transactions in Operational Research*. IN SUBMISSION

Preliminary results were presented at:

Monroy-Licht, M., Amaya, C.A., Langevin, Rousseau, L-M. (2015). The rescheduling capacitated arc routing problem. *CORS/INFORMS International Conference*. June 14-17, Montréal, Canada

CHAPTER 6 GENERAL DISCUSSION

We have studied two problems of road winter maintenance: i) scheduling of a time-sensitive route for detection of black-ice over roads, and ii) rescheduling of routes for snow plowing or salt spreading after vehicles failure. To do this, we presented arc routing models and optimization solution methods for both cases.

The road network monitoring for black-ice detection for a single vehicle was modeled as a RPPTW. We proposed different mixed integer linear programming formulations, an exact method and a heuristic approach to solve the problem.

We started exploring the “direct” formulation of the CARPTW given by Gueguen (1999). However even if we introduce a new set of constraints that reduces the number of equivalent solutions, this formulation is still not practical. Therefore we turned to two formulations that transform the original graph: “model on edges” and “model on nodes”. The transformation used shortest path algorithms to produce the travel time on the edges in the transformed graph. The formulations were tested on a set of generated instances with different width of time windows. As expected, the performance of formulations in terms of time and number of problems solved to optimality is better for instances with tight time windows.

Then, we moved toward an exact solution method. Our choice is a polyhedral approach, which was already used by Letchford and Eglese (1998) for the RPP with deadline classes. We proposed a cutting plane algorithm for the “model on the nodes”, which is equivalent to a TSP with time windows and side constraints. This approach was tested on two sets of instances: the set of generated instances and a set of instances obtained from a real network. The first set of instances has a different time window for each required edge while the second set includes 4 or 5 time slots defining the time windows; therefore subsets of required edges share the same time windows. The cutting plane algorithm was able to solve to optimality all the instances based on the real network, which are larger than the generated instances, in less than 11 minutes. The algorithm solved to optimality most of the generated instances, but the computational times for the harder instances of this group are high, reaching in some cases around 7 hours. This shows that the cutting plane algorithm works better for time windows structured by time slots.

Other valid inequalities for the RRPTW could improve the efficiency of the method, especially inequalities that consider precedence constraints (Balas et al., 1995). The predecessor/successor inequalities were the most valuable in our tests.

A second approach to solve the RPPTW is presented in Chapter 4. We implemented an ALNS algorithm, which aims at finding good quality solutions in shorter time for the larger instances. Several versions of the algorithm were tested to compare the performance of seven removal and two insertion heuristics and a full version that includes all of them. The best version of the metaheuristic showed an efficient performance: for the hardest generated instances (wide time windows) the average gap was 0.17% while the computational time reported by the cutting plane algorithm was significantly reduced. In the case of the instances based on the real network, the metaheuristic found 6 optimal solutions out of 8 problems and the gap reported for the other two problems was less than 0.6%. However the running times of the exact method in this set of instances were on average better.

The quality of initial solution in the performance of the metaheuristic could have an impact in terms of computational time: we allow infeasible initial solutions, and therefore the ALNS algorithm may take more time to converge to a good solution. In addition, the problems of time windows given by time slots may suggest using other heuristics that remove and insert requests with different rules to those proposed here.

The Chapter 5 of this thesis focuses on the problem of rescheduling an initial itinerary of road maintenance operations such as snow plowing or salt spreading when a vehicle breaks down. In this case various vehicles execute the initial plan and suddenly one or more vehicle failures occur. Different policies considering operational and disruption costs are analyzed in the rerouting phase: mixed integer programming formulations are presented to model the policies.

We have supposed that active vehicles could exceed the remaining capacity in order to service all the required edges in the rerouting phase. We do not consider extra costs if a vehicle should be refilled in the case of salt spreading or extra costs if drivers exceed the duration of their shifts. The operational costs reflect only traveled distances in the rerouting phase. On the other hand, we determined a metric of disruption costs given by the number of edges moved to a different route in the rescheduling.

We generated test instances from the benchmark CARP instances and larger instances from a real network. Our tests consider simulations of early, middle, and late vehicle breakdown during the shifts. When a vehicle breakdown occurs early the negative impact on the extra operational costs is lower than with middle or late vehicle breakdown. On an early vehicle breakdown there are more available resources close to the failure zone, contrary to the other cases, where the resources could be far away from the failure zone and it might be expensive to move resources to the failure zone.

We showed that minimizing the operational and disruption costs are objectives in conflict and we presented a solution strategy that offers a set of solutions to choose from, depending on the policy of the planner. The strategy is based on the idea of fixing some decision variables to their values in the initial schedule in order to reduce the disruption costs, and then a smaller problem is solved using one of the mixed integer programming formulations proposed. This method gives quick response to the planner in a dynamic context when larger instances are considered.

At the end of Chapter 5 we compared our disruption metric to 4 metrics already used in optimizations problems, which measure the similarity of solutions encoded by permutations. We calculated the values of these metrics for our solutions and we estimated the bounds of these metrics based on the best possible case. In general, our solutions are relatively far from the bounds of each metric however we believe that other possible measures of disruption costs may arise from an adaptation of these metrics to the routing context.

CHAPTER 7 CONCLUSIONS AND RECOMMENDATIONS

This dissertation has contributed to the development of arc routing models and algorithms for road network monitoring and rescheduling of snow plowing and salt spreading operations. In the first case a *rural postman problem with time windows* (RPPTW) is explored and a *rescheduling capacitated arc routing problem* (RCARP) is presented in the second case.

The RPPTW and the RCARP consider time-sensitive and dynamic features respectively; both aspects are important in real applications of road winter maintenance operations. The former problem looks at minimizing the total traveled cost; this dissertation presents for the first time solution methods with numerical results for this problem. The second problem is introduced in this work; due to the application context we consider operational and disruption costs to be minimized.

The RPPTW were attacked from two optimization approaches: exact and heuristic solutions methods were implemented to solve the problem.

In this work, we have presented three models for the RPPTW. We modified the Gueguen's formulation (1999) for the *capacitated arc routing problem with time windows* (CARPTW) and we proposed a new set of constraints that allows reducing the number of equivalent solutions. We have also presented two models for the RPPTW built on transformed graphs using the shortest path assumption. All the models can be solved by integer optimization methods, and their limitation is naturally the size of the instance to solve. The latter two models were solved for 225 randomly generated instances with different width of time windows and up to 45 required edges. Computational results showed that the "model on the nodes" performs better than the "model on the edges": the former solved to optimality twelve more instances and its computational times were smaller.

We have proposed a cutting plane algorithm for the "model on the nodes". We included valid inequalities derived from the valid inequalities for the *traveling salesman problem* (TSP) *with time windows* and the *precedence constrained* TSP. The cutting plane algorithm was tested on a set of 225 randomly generated instances and on a set of larger instances based on the real network of the "Estrie" region in Quebec; these instances have 140 nodes and 104 required edges.

The cutting plane algorithm was able to solve 224 of 225 instances from the first group and all the instances based on the real network. 222 instances from the first set were solved in less than 2 hours, while instances from the second set were solved in less than 11 minutes. The first set of generated instances has a different time windows for each required edge, whereas the second set of instances has time windows structured by time slots, therefore subsets of required edges have the same time windows. Even if instances are larger in terms of required edges, it seems more difficult to solve problems with a smaller number of required edges but with time windows structured by each required edge.

We have also explored a heuristic solution method to the RPPTW. An *adaptive large neighborhood search* (ALNS) algorithm is presented. We proposed seven removal heuristics and two insertion heuristics which were tested in different versions of the ALNS algorithm. The best version of the algorithm called VR124I12 includes: i) two removal heuristics based on randomness and one that considers distance requirements as rules to remove requests, and ii) the two insertion heuristics proposed (basic greedy insertion and regret insertion) to rebuild a partial solution.

The computation results have showed that our ALNS algorithm performs well, solving to optimality 224 of the 232 instances presented previously while significantly reducing the computational time reported by the cutting plane algorithm on the hardest instances.

There are several ways that the cutting plane can be improved as well as the ALNS algorithm. Other types of valid inequalities and the inclusion of heuristics that find good upper bounds could help the cutting plane become more competitive. A natural extension is to develop a branch and cut algorithm that aims at solving larger instances. On the other hand, new removal and insertion heuristics could improve the efficiency of the ALNS algorithm taking into account the structure of the time windows of the problem. Finally, the inclusion of other constraints in the RPPTW as for example the prohibition of U-turns, for safety reasons, and the extension to the case of multiple vehicles are possible areas of research.

Regarding the RCARP we have made a contribution in the field of dynamic arc routing problems. We consider the rescheduling of the CARP when a vehicle fails and the initial plan must be modified. We have introduced the problem exploring different policies in the rerouting phase and we have proposed a solution strategy based on mixed integer programming.

We have modeled four policies that consider minimizing operational cost and disruption. We compared the policies and concluded that the concerned costs are objectives in conflict. Our computational tests showed that the model that minimizes both costs (O4) works well, getting a good trade-off between the conflicting objectives.

We have presented a heuristic strategy for obtaining quickly solutions to the problem. The idea considers a smaller problem to solve because some decision variables are fixed, and then the smaller problem is solved to optimality using one of the four formulations proposed. The strategy was tested on a set of instances generated from a real network having up to 88 required edges and 10 active vehicles after the vehicle breakdown. The strategy showed to be efficient solving the two largest problems in less than one minute for 16 combinations of parameters "*per*" and "*model*". The limitation of the method is given by the size of the instance to solve (we used CPLEX to solve the formulations).

In addition, we have explored other possible metrics of disruption costs based on measures of distance between permutations. However, these metrics need to be adapted for the routing context.

This dynamic problem requires quick responses; therefore the implementation of multiobjective metaheuristics that find good quality solutions in short time can be a future area of research. Future work will consider other constraints as balance of routes and the addition of penalties when the remaining capacities are exceeded.

REFERENCES

Afsar, H. M. (2010). A Branch-and-Price Algorithm for Capacitated Arc Routing Problem with Flexible Time Windows. *Electronic Notes in Discrete Mathematics, 36*(0), 319-326. doi: http://dx.doi.org/10.1016/j.endm.2010.05.041

Alidaee, B., & Womer, N. K. (1999). Scheduling with time dependent processing times: Review and extensions. *Journal of the Operational Research Society, 50*(7), 711-720. doi: 10.1057/palgrave.jors.2600740

Amaya, A., Langevin, A., & Trépanier, M. (2007). The capacitated arc routing problem with refill points. *Operations Research Letters, 35*(1), 45-53. doi: http://dx.doi.org/10.1016/j.orl.2005.12.009

Amaya, C. A., Langevin, A., & Trepanier, M. (2010). A heuristic method for the capacitated arc routing problem with refill points and multiple loads. *J Oper Res Soc, 61*(7), 1095-1103.

Aminu, U. F., & Eglese, R. W. (2006). A constraint programming approach to the Chinese postman problem with time windows. *Computers & Operations Research, 33*(12), 3423-3431. doi: http://dx.doi.org/10.1016/j.cor.2005.02.012

Ascheuer, N., Fischetti, M., & Grötschel, M. (1999). Solving the Asymmetric Traveling Salesman Problem with Time Windows by branch-and-cut. Belin, Germany: Konrad-Zuse-Zentrun für informationstechnik Berlin.

Ascheuer, N., Fischetti, M., & Grötschel, M. (2001). Solving the Asymmetric Travelling Salesman Problem with time windows by branch-and-cut. *Mathematical Programming, 90*(3), 475-506. doi: 10.1007/PL00011432

Ascheuer, N., Jünger, M., & Reinelt, G. (2000). A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints. *Computational Optimization and Applications, 17*(1), 61-84. doi: 10.1023/A:1008779125567

Balas, E., Fischetti, M., & Pulleyblank, W. (1995). The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming, 68*(1-3), 241-265. doi: 10.1007/BF01585767

Belenguer, E., & Benavent, E. (1991). Polyhedral results on the capacitated arc routing problem. Spain: Departamento de Estadística e Investigación Operativa, Universidad de Valencia.

Benavent, E., Campos, V., Corberan, A., & Mota, E. (1992). The Capacitated Chinese Postman Problem: Lower bounds. *Networks, 22*, 669-690.

Beullens, P., Muyldermans, L., Cattrysse, D., & Van Oudheusden, D. (2003). A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research, 147*(3), 629-643. doi: http://dx.doi.org/10.1016/S0377-2217(02)00334-X

Bode, C. (2014). Personal communication, July 18, 2014.

Brandão, J., & Eglese, R. (2008). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research, 35*(4), 1112-1126. doi: http://dx.doi.org/10.1016/j.cor.2006.07.007

Busch, I. K. (1991). *Vehicle routing on acyclic networks.* (Ph.D. Dissertation), The Johns Hopkins University, Baltimore, Md.

Campbell, J. F., & Langevin, A. (2000). Roadway Snow and Ice Control. In M. Dror (Ed.), *Arc Routing* (pp. 389-418): Springer US.

Campbell, J. F., Langevin, A., & Perrier, N. (2014). Advances in vehicle routing for snow plowing. In Á. Corberán & G. Laporte (Eds.), *Arc Routing: Problems, Methods, and Applications.* (pp. 321-350): SIAM.

Christofides, N. (1973). The optimum traversal of a graph. *Omega, 1*, 13.

Christofides, N., Campos, V., Corberán, Á., & Mota, E. (1981). An algorithm for the Rural Postman Problem. London: Imperial College.

Clarke, G., & Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations research, 12*(4), 568-581. doi: doi:10.1287/opre.12.4.568

Corberán, A., & Prins, C. (2010). Recent results on Arc Routing Problems: An annotated bibliography. *Networks, 56*(1), 50-69. doi: 10.1002/net.20347

Corberán, A., & Sanchis, J. M. (1994). A polyhedral approach to the rural postman problem. *European Journal of Operational Research, 79*(1), 95-114. doi: http://dx.doi.org/10.1016/0377-2217(94)90398-0

Corberán, Á., & Sanchis, J. M. (1991). A polyhedral approach to the Rural Postman Problem (D. d. E. e. I. operative, Trans.). Spain: Universidad de Valencia.

Desrochers, M. (1988). An algorithm for the shortest path problem with resource constraints *Cahiers du GERARD*. Montreal, Canada: Ecole des Hautes Etudes Commerciales.

Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters, 10*(1), 27-36. doi: http://dx.doi.org/10.1016/0167-6377(91)90083-2

Desrosiers, J., Dumas, Y., Solomon, M. M., & Soumis, F. (1995). Time constrained routing and scheduling. *Handbooks in operations research and management science, 8*, 35-139.

Dror, M. (2000). *Arc routing: theory, solutions and applications*: Kluwer Academic Publishers.

Dror, M., Leung, J. Y., & Mullaseril, P. (2000). Livestock Feed Distribution and Arc Traversal Problems. In M. Dror (Ed.), *Arc Routing* (pp. 443-464): Springer US.

Edmonds, J., & Johnson, E. (1973). Matching, Euler tours and Chinese postman. *Mathematical Programming, 5*, 36.

Eglese, R., Golden, B., Wasil, E. (2014). Route optimization for meter reading and salt spreading. In Ángel Corberán & Gilbert Laporte (eds.) *Arc Routing: Problems, Methods, and Applications*, 303-320.

Eglese, R., & Letchford, A. (2000). Polyhedral Theory for Arc Routing Problems. In M. Dror (Ed.), *Arc Routing* (pp. 199-230): Springer US.

Eglese, R. W. (1994). Routeing winter gritting vehicles. *Discrete Applied Mathematics, 48*(3), 231-244. doi: http://dx.doi.org/10.1016/0166-218X(92)00003-5

Eiselt, H. A., Gendreau, M., & Laporte, G. (1995). Arc Routing Problems, Part II: The Rural Postman Problem. *Operations research, 43*(3), 399-414. doi: doi:10.1287/opre.43.3.399

Fischetti, M., & Toth, P. (1997). A Polyhedral Approach to the Asymmetric Traveling Salesman Problem. *Management Science, 43*(11), 1520-1536. doi: 10.2307/2634585

Fu, L., Trudel, M., & Kim, V. (2009). Optimizing winter road maintenance operations under real-time information. *European Journal of Operational Research, 196*(1), 332-341. doi: http://dx.doi.org/10.1016/j.ejor.2008.03.001

Golbaharan, N. (2001). *An Application of Optimization to the Snow Removal Problem: A Column Generation Approach*: Division of Optimization, Department of Mathematics, Linköpings university.

Golden, B. L., Dearmon, J. S., & Baker, E. K. (1983). Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research, 10*(1), 47-59. doi: http://dx.doi.org/10.1016/0305-0548(83)90026-6

Golden, B. L., & Wong, R. T. (1981). Capacitated arc routing problems. *Networks, 11*(3), 305-315. doi: 10.1002/net.3230110308

Gueguen, C. (1999). *Méthodes de résolution exacte pour les problèmes de tournées de véhicules.* École Central Paris.

Handa, H., Chapman, L., & Xin, Y. (2005, 5-5 Sept. 2005). *Dynamic salting route optimisation using evolutionary computation.* Paper presented at the Evolutionary Computation, 2005. The 2005 IEEE Congress.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., & Yagiura, M. (2005). Effective Local Search Algorithms for Routing and Scheduling Problems with General Time-Window Constraints. *Transportation Science, 39*(2), 206-232. doi: doi:10.1287/trsc.1030.0085

Johnson, E. L., & Wøhlk, S. (2009). solving the capacitated arc routing problem with time windows using column generation (D. o. B. Studies, Trans.) *Coral working papers*: University of Aarhus.

Johnson, E. L., & Wøhlk, S. (2009). Solving the capacitated arc routing problem with time windows using column generation: University of Aarhus, Aarhus School of Business, Department of Business Studies.

Jünger, M., Reinelt, G., & Rinaldi, G. (1995). Chapter 4 The traveling salesman problem. In T. L. M. C. L. M. M.O. Ball & G. L. Nemhauser (Eds.), *Handbooks in operations research and management science* (Vol. Volume 7, pp. 225-330): Elsevier.

Kang, M.-J., & Han, C.-G. (1998). Comparison of Crossover Operators for Rural Postman Problem with Time Windows. In P. K. Chawdhry, R. Roy, & R. K. Pant (Eds.), *Soft Computing in Engineering Design and Manufacturing* (pp. 259-267): Springer London.

Kelleher, G., & Cavichiollo, P. (1999). *Intelligent support of the Rescheduling of complex Manufacturing domains – an Example application.* Paper presented at the 2da international workshop on IMS, Leuven, Belgium.

Kwan, M.-K. (1962). Graphic programming using odd or even points. *Chinese Mathematics, 1*, 5.

Lenstra, J. K., & Kan, A. H. G. R. (1976). On general routing problems. *Networks, 6*(3), 273-280. doi: 10.1002/net.3230060305

Letchford, A. N., & Eglese, R. W. (1998). The rural postman problem with deadline classes. *European Journal of Operational Research, 105*(3), 390-400. doi: http://dx.doi.org/10.1016/S0377-2217(97)00090-8

Levenshtein, V. I. (1966). *Binary codes capable of correcting deletions, insertions, and reversals.* Paper presented at the Soviet physics doklady.

Li, J.-Q., Borenstein, D., & Mirchandani, P. B. (2007). A decision support system for the single-depot vehicle rescheduling problem. *Computers & Operations Research, 34*(4), 1008-1032. doi: http://dx.doi.org/10.1016/j.cor.2005.05.022

Li, J.-Q., Mirchandani, P. B., & Borenstein, D. (2004). *Parallel auction algorithm for bus rescheduling.* Paper presented at the Proceedings of the Ninth International Conference on Computer-Aided Scheduling of Public Transport, San Diego, California, USA.

Li, J.-Q., Mirchandani, P. B., & Borenstein, D. (2009). A Lagrangian heuristic for the real-time vehicle rescheduling problem. *Transportation Research Part E: Logistics and Transportation Review, 45*(3), 419-433. doi: http://dx.doi.org/10.1016/j.tre.2008.09.002

Li, L. Y. O. (1992). *Vehicle routeing for winter gritting.* (Ph.D. Thesis), Lancaster University, UK.

Li, L. Y. O., & Eglese, R. W. (1996). An Interactive Algorithm for Vehicle Routeing for Winter Gritting. *J Oper Res Soc, 47*(2), 217-228.

Liu, G., Ge, Y., Qiu, T. Z., & Soleymani, H. R. (2014). Optimization of snow plowing cost and time in an urban environment: A case study for the City of Edmonton. *Canadian Journal of Civil Engineering, 41*(7), 667-675. doi: doi:10.1139/cjce-2013-0409

Liu, M. (2014, September 16). *A memetic algorithm with a new split scheme for solving dynamic capacitated arc routing problems.* Paper presented at the Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014.

Liu, M., Kumar, S., & Ray, T. (2014, July 6-11, 2014). *A memetic algorithm with a new split scheme for solving dynamic capacitated arc routing problems.* Paper presented at the Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China.

Martí, R., Laguna, M., & Campos, V. (2005). Scatter Search vs. Genetic Algorithms. In R. Sharda, S. Voß, C. Rego, & B. Alidaee (Eds.), *Metaheuristic Optimization via Memory and Evolution* (Vol. 30, pp. 263-282): Springer US.

Marzolf, F., Trépanier, M., & Langevin, A. (2006). Road network monitoring: algorithms and a case study. *Computers & Operations Research, 33*(12), 3494-3507. doi: http://dx.doi.org/10.1016/j.cor.2005.02.040

Min, L., Singh, H. K., & Ray, T. (2014, 6-11 July 2014). *A benchmark generator for dynamic capacitated arc routing problems.* Paper presented at the Evolutionary Computation (CEC), 2014 IEEE Congress.

Monroy-Licht, M., Amaya, C., & Langevin, A. (2015). Adaptive Large Neighborhood Search for the Rural Postman Problem with Time Windows. Montréal: Cirrelt.

Monroy-Licht, M., Amaya, C. A., & Langevin, A. (2014). The Rural Postman Problem with time windows. *Networks, 64*(3), 169-180. doi: 10.1002/net.21569

Monroy, I. M., Amaya, C. A., & Langevin, A. (2013). The periodic capacitated arc routing problem with irregular services. *Discrete Applied Mathematics, 161*(4–5), 691-701. doi: http://dx.doi.org/10.1016/j.dam.2011.05.014

Moreira, L. M., Oliveira, J. F., Gomes, A. M., & Ferreira, J. S. (2007). Heuristics for a dynamic rural postman problem. *Computers & Operations Research, 34*(11), 3281-3294. doi: http://dx.doi.org/10.1016/j.cor.2005.12.008

Mu, Q., Fu, Z., Lysgaard, J., & Eglese, R. (2011). Disruption management of the vehicle routing problem with vehicle breakdown. *J Oper Res Soc, 62*(4), 742-749.

Mullaseril, P. A. (1997). *Capacitated rural postman problem with time windows and split delivery.* (PhD.), University of Arizona, Arizona.

Mullaseril, P. A., & Dror, M. (1996). A set covering approach for directed node and arc routing problems with split deliveries and time windows: MIS Department, University of Arizona.

Mullaseril, P. A., Dror, M., & Leung, J. (1997). Split-Delivery Routeing Heuristics in Livestock Feed Distribution. *The Journal of the Operational Research Society, 48*(2), 107-116. doi: 10.2307/3010350

Nagata, Y. (1997). Edge Assembly Crossover.A High-power Genetic Algorithm for the Traveling Salesman Problem. *Proc. 7th ICGA,* 450-457.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and Combinatorial Optimisation*. New York: Wiley.

Nobert, Y., & Picard, J. C. (1994). *A heuristic algorithm for the Rural Postman Problem with Time Windows.* Paper presented at the ORSA/TIMS, Detroit.

Office of the Auditor General of Ontario. (2015). Winter highway maintenance *Special report* (pp. 44).

Orda, A., & Rom, R. (1990). Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *J. ACM, 37*(3), 607-625. doi: 10.1145/79147.214078

Orloff, C. S. (1976). On general routing problems: Comments. *Networks, 6*(3), 281-284. doi: 10.1002/net.3230060306

Padberg, M., & Rinaldi, G. (1990). An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming, 47*(1-3), 19-36. doi: 10.1007/BF01580850

Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev., 33*(1), 60-100. doi: 10.1137/1033004

Papadimitriou, C. H. (1976). On the complexity of edge traversing. *Journal of ACM, 23*, 10.

Pearn, W.-L., Assad, A., & Golden, B. L. (1987). Transforming arc routing into node routing problems. *Computers & Operations Research, 14*(4), 285-288.

Perrier, N., Campbell, J. F., Gendreau, M., & Langevin, A. (2012). Vehicle Routing Models and Algorithms for Winter Road Spreading Operations. In R. M.-T. Jairo, A. J. Angel, H. Luisa Huaccho, F. Javier, & L. R.-V. Gloria (Eds.), *Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing and Scheduling Solutions* (pp. 15-45). Hershey, PA, USA: IGI Global.

Perrier, N., Langevin, A., & Campbell, J. F. (2006a). A survey of models and algorithms for winter road maintenance. Part I: system design for spreading and plowing. *Computers & Operations Research, 33*(1), 209-238. doi: http://dx.doi.org/10.1016/j.cor.2004.07.006

Perrier, N., Langevin, A., & Campbell, J. F. (2006b). A survey of models and algorithms for winter road maintenance. Part II: system design for snow disposal. *Computers & Operations Research, 33*(1), 239-262. doi: http://dx.doi.org/10.1016/j.cor.2004.07.007

Perrier, N., Langevin, A., & Campbell, J. F. (2007a). A survey of models and algorithms for winter road maintenance. Part III: Vehicle routing and depot location for spreading. *Computers & Operations Research, 34*(1), 211-257. doi: http://dx.doi.org/10.1016/j.cor.2005.05.007

Perrier, N., Langevin, A., & Campbell, J. F. (2007b). A survey of models and algorithms for winter road maintenance. Part IV: Vehicle routing and fleet sizing for plowing and snow disposal. *Computers & Operations Research, 34*(1), 258-294. doi: http://dx.doi.org/10.1016/j.cor.2005.05.008

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225*(1), 1-11. doi: http://dx.doi.org/10.1016/j.ejor.2012.08.015

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34*(8), 2403-2435. doi: http://dx.doi.org/10.1016/j.cor.2005.09.012

Prins, C., Lacomme, P., & Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies, 40*(0), 179-200. doi: http://dx.doi.org/10.1016/j.trc.2014.01.011

Razmara, G. (2004). *Snow removal routing problems: theory and applications.* (PhD.), Linköping University, Linköping, Sweden.

Reghioui, M., Prins, C., & Labadi, N. (2007). GRASP with Path Relinking for the Capacitated Arc Routing Problem with Time Windows. In M. Giacobini (Ed.), *Applications of Evolutionary Computing* (Vol. 4448, pp. 722-731): Springer Berlin Heidelberg.

Rhalibi, A., & Kelleher, G. (2003, 5-8 Oct. 2003). *An approach to dynamic vehicle routing, rescheduling and disruption metrics.* Paper presented at the Systems, Man and Cybernetics, 2003. IEEE International Conference.

Riquelme-Rodríguez, J.-P., Langevin, A., & Gamache, M. (2014). Adaptive large neighborhood search for the periodic capacitated arc routing problem with inventory constraints. *Networks, 64*(2), 125-139. doi: 10.1002/net.21562

Rodrigues, A. (2013). Edmonton spends more on snow removal than any other city in Western Canada, figures show. *Edmonton sun*. Retrieved from http://www.edmontonsun.com/2013/01/10/edmonton-spends-more-on-snow-removal-than-any-other-city-in-western-canada-figures-show#disqus_thread website

Ronald, S. (1998, 4-9 May 1998). *More distance functions for order-based encodings.* Paper presented at the Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence.

Ropke, S., & Pisinger, D. (2006a). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science, 40*(4), 455-472. doi: 10.1287/trsc.1050.0135

Ropke, S., & Pisinger, D. (2006b). A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research, 171*(3), 750-775. doi: http://dx.doi.org/10.1016/j.ejor.2004.09.004

Salazar-Aguilar, M. A., Langevin, A., & Laporte, G. (2012). Synchronized arc routing for snow plowing operations. *Computers & Operations Research, 39*(7), 1432-1440. doi: http://dx.doi.org/10.1016/j.cor.2011.08.014

Salazar-Aguilar, M. A., Langevin, A., & Laporte, G. (2013). The synchronized arc and node routing problem: Application to road marking. *Computers & Operations Research, 40*(7), 1708-1715. doi: http://dx.doi.org/10.1016/j.cor.2013.01.007

Slone, S. (2014). High costs of winter road maintenance, 2013-14. *Capitol research,* 4. Retrieved from http://knowledgecenter.csg.org/kc/system/files/CR_WinterMaintenanceCosts.pdf website

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research, 35*(2), 254-265.

Sörensen, K. (2006). Route stability in vehicle routing decisions: a bi-objective approach using metaheuristics. *Central European Journal of Operations Research, 14*(2), 193-207. doi: 10.1007/s10100-006-0168-3

Springintveld, J., Vaandrager, F., & R. D'Argenio, P. (2001). Testing timed automata. *Theoretical Computer Science, 254*(1–2), 225-257. doi: http://dx.doi.org/10.1016/S0304-3975(99)00134-6

Sun, J., Meng, Y., & Tan, G. (2013). A Cutting Plane Heuristic Algorithm for the Time Dependent Chinese Postman Problem. In M. Fellows, X. Tan, & B. Zhu (Eds.), *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management* (Vol. 7924, pp. 163-174): Springer Berlin Heidelberg.

Sun, J., Meng, Y., & Tan, G. (2015). An integer programming approach for the Chinese postman problem with time-dependent travel time. *Journal of Combinatorial Optimization, 29*(3), 565-588. doi: 10.1007/s10878-014-9755-8

Sun, J., Tan, G., & Meng, X. (2011, 19-22 Aug. 2011). *Graph transformation algorithm for the time dependent Chinese Postman Problem with Time Windows.* Paper presented at the Mechatronic Science, Electric Engineering and Computer (MEC).

Sundararaghavan, P. S., & Kunnathur, A. S. (1994). Single machine scheduling with start time dependent processing times: Some solvable cases. *European Journal of Operational Research, 78*(3), 394-403. doi: http://dx.doi.org/10.1016/0377-2217(94)90048-5

Tagmouti, M., Gendreau, M., & Potvin, J.-Y. (2007). Arc routing problems with time-dependent service costs. *European Journal of Operational Research, 181*(1), 30-39. doi: http://dx.doi.org/10.1016/j.ejor.2006.06.028

Tagmouti, M., Gendreau, M., & Potvin, J.-Y. (2010). A variable neighborhood descent heuristic for arc routing problems with time-dependent service costs. *Computers & Industrial Engineering, 59*(4), 954-963. doi: http://dx.doi.org/10.1016/j.cie.2010.09.006

Tagmouti, M., Gendreau, M., & Potvin, J.-Y. (2011). A dynamic capacitated arc routing problem with time-dependent service costs. *Transportation Research Part C: Emerging Technologies, 19*(1), 20-28. doi: http://dx.doi.org/10.1016/j.trc.2010.02.003

Tan, G., & Sun, J. (2011). An Integer Programming Approach for the Rural Postman Problem with Time Dependent Travel Times. In B. Fu & D.-Z. Du (Eds.), *Computing and Combinatorics* (Vol. 6842, pp. 414-431): Springer Berlin Heidelberg.

Tan, G., Sun, J., & Hou, G. (2013). The time-dependent rural postman problem: polyhedral results. *Optimization Methods and Software, 28*(4), 855-870. doi: 10.1080/10556788.2012.666240

Transportation Research Board (2005). *Winter highway operations, A synthesis of higway practice*. Washington DC: Transportation Research Board.

Ulusoy, G. (1985). The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research, 22*(3), 329-337. doi: http://dx.doi.org/10.1016/0377-2217(85)90252-8

Wang, H.-F., & Wen, Y.-P. (2002). Time-constrained Chinese postman problems. *Computers & Mathematics with Applications, 44*(3–4), 375-387. doi: http://dx.doi.org/10.1016/S0898-1221(02)00156-6

Weise, T., Devert, A., & Tang, K. (2012). *A developmental solution to (dynamic) capacitated arc routing problems using genetic programming*. Paper presented at the Proceedings of the 14th annual conference on Genetic and evolutionary computation, Philadelphia, Pennsylvania, USA.

Wøhlk, S. (2005). *Contributions to arc routing*. University of Southern Denmark.

Yazici, A., Kirlik, G., Parlaktuna, O., & Sipahioglu, A. (2014). A Dynamic Path Planning Approach for Multirobot Sensor-Based Coverage Considering Energy Constraints. *Cybernetics, IEEE Transactions on, 44*(3), 305-314. doi: 10.1109/TCYB.2013.2253605